

OLIVER MONTENBRUCK
THOMAS PFLEGER

Astronomie mit dem Personal Computer

Dritte Auflage



Springer



Montenbruck · Pfleger
Astronomie mit dem Personal Computer

Springer-Verlag Berlin Heidelberg GmbH

Oliver Montenbruck Thomas Pfleger

Astronomie mit dem Personal Computer

Dritte, völlig neubearbeitete Auflage
mit 46 Abbildungen und einer



Springer

Dr. rer. nat. Oliver Montenbruck
DLR-GSOC, D-82230 Weßling
e-mail: oliver.montenbruck@dlr.de

Dipl.-Ing. Thomas Pfleger
Schützenstraße 25, D-53773 Hennef
e-mail: th.pfleger@t-online.de

Umschlagbild: Der Komet Hale-Bopp, aufgenommen am 29.03.1997 in Potzlow bei Seehausen.
Links oben im Bild ist der Sternhaufen $h + \chi$ im Perseus zu sehen, rechts unten der Andromedanebel
(M 31). Die Belichtungszeit betrug 10 min.
Bildarchiv von Sterne und Weltraum; Bildautor: Thomas Helms.

Additional material to this book can be downloaded from <http://extras.springer.com>.

ISBN 978-3-662-05863-3 ISBN 978-3-662-05862-6 (eBook)
DOI 10.1007/978-3-662-05862-6

Die Deutsche Bibliothek – CIP-Einheitsaufnahme
Astronomie mit dem Personal Computer [Medienkombination] / Oliver Montenbruck ; Thomas Pfleger. –
Berlin ; Heidelberg; New York; Barcelona; Hongkong; London; Mailand; Paris; Singapur; Tokio; Springer
Engl. Ausg. u.d.T.: Astronomy on the personal computer

Buch. – 3., völlig neubearb. Aufl. – 1999
CD-ROM. – 3., völlig neubearb. Aufl. – 1999

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwidderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.

© Springer-Verlag Berlin Heidelberg 1989, 1994, 1999
Ursprünglich erschienen bei Springer-Verlag Berlin Heidelberg New York 1999.

Softcover reprint of the hardcover 3rd edition 1999

Datenkataloge auf der CD-ROM:

PPM Star Catalogue, Volume I & II
© Astronomisches Rechen-Institut Heidelberg, Germany 1991
PPM Star Catalogue, Volume III & IV
© Astronomisches Rechen-Institut Heidelberg, Germany 1993

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, daß solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Satz: Reproduktionsfertige Vorlagen von den Autoren
Einbandgestaltung: Erich Kirchner, Heidelberg

SPIN: 10732764 55/3144/ba - 5 4 3 2 1 0 - Gedruckt auf säurefreiem Papier

Vorwort zur dritten Auflage

Blickt man auf die vergangenen zehn Jahre zurück, so läßt sich zunächst beruhigt feststellen, daß die astronomischen und mathematischen Verfahren unseres Buches „*Astronomie mit dem Personal Computer*“ auch kurz vor Ausgang des Jahrtausends wenig von ihrer Gültigkeit und Aktualität eingebüßt haben. Beeindruckend ist dagegen die stürmische, ja explosionsartige, Entwicklung der Computertechnik, die sich bereits in vielen Haushalten einen festen Platz neben Telefon und Fernseher erobert hat.

Es ist daher nicht verwunderlich, daß wir uns für die vorliegende Neuauflage auf eine gründliche Überarbeitung und Modernisierung aller Programme konzentriert haben. Angesichts der immer kleineren Verbreitung der Programmiersprache PASCAL sind wir gerne dem vielfach geäußerten Wunsch gefolgt, unsere Programme in C++ zur Verfügung zu stellen. Sieht man einmal von systemspezifischen Sprachen wie VISUALBASIC ab, so stellen C und C++ die wohl zur Zeit populärsten und meistgenutzten Programmiersprachen dar. Neben der Verfügbarkeit auf den verschiedensten Computersystemen leitet sich ihre Bedeutung auch aus der guten Unterstützung systemnaher Anwendungen ab. Dies gilt insbesondere für C++, das durch objektorientierte Programmietechniken besonders gut zur Entwicklung graphischer Benutzeroberflächen geeignet ist.

Bei der Anpassung der bisherigen PASCAL Programme haben wir uns nicht auf eine reine Übersetzung beschränkt, sondern eine vollständige Neuimplementierung vorgenommen, die den vielfältigen Möglichkeiten von C++ Rechnung trägt. Wir hoffen, dem Leser damit ein modernes und leistungsfähiges Paket von Programmen und Bibliotheken zur Verfügung zu stellen, ohne dabei die Verständlichkeit der in den vorangehenden Auflagen bewährten Darstellung zu opfern.

Inhaltliche Änderungen ergeben sich unter anderem aus der konsequenten Einführung der Vektor- und Matrix-Schreibweise, die eine kompakte und zeitgemäße Darstellung aller Koordinatentransformationen erlaubt und in den C++ Programmen durch entsprechende Klassen und Operatoren abgebildet wird. So weit erforderlich wurden Anpassungen an aktuelle Konventionen und Zahlenwerte vorgenommen. Dies betrifft zum einen verbesserte Rotationselemente der großen Planeten, zum anderen die Zählung der geographischen Länge der Erde. Wie in der Geographie und Satellitengeodäsie wird diese mittlerweile auch in der Astronomie nach Osten positiv gezählt. Hieraus ergeben sich z. B. Vorzeichenunterschiede bei der Berechnung der Ortssternzeit oder bei den Stationskoeffizienten von Sternbedeckungen, die in den einzelnen Algorithmen und Programmen entsprechend berücksichtigt sind.

Das Kapitel über die Bahnbestimmung aus drei Beobachtungen wurde um die Darstellung der vollständigen Gaußschen Bahnbestimmungsmethode erweitert, so daß nun auch Fälle mit mehrfachen Lösungen problemlos behandelt werden können. Neu hinzugekommen ist das Programm **Phases**, das neben der Berechnung von Mondphasen mögliche Sonnen- und Mondfinsternisse identifiziert. An vielen Stellen wurden Verbesserungen vorgenommen, um die Programme anwendungsfreundlicher zu gestalten. So kann z.B. bei der Berechnung von Auf- und Untergängen die Definition der Dämmerung (bürgerlich, nautisch, astronomisch) frei gewählt werden und bei der Berechnung von Sternbedeckungen werden schlecht beobachtbare Ereignisse anhand umfangreicher Kriterien unterdrückt. So weit erforderlich, können Ein- und Ausgabedateien bei allen Programmen in der Kommandozeile festgelegt werden.

Mit der beigelegten CD ergibt sich erstmals die Möglichkeit, auch die ausführbaren Programme für PCs unter Windows und Linux in unkomprimierter Form zur Verfügung zu stellen. Somit können auch Leser ohne C++ Compiler oder Programmiererfahrung die CD mit ihren Daten und Programmen sofort nutzen und für eigene Berechnungen einsetzen. Die zusätzliche Anschaffung eines Compilers ist lediglich für diejenigen Leser erforderlich, die die einzelnen Programme an ihre persönlichen Bedürfnisse anpassen oder erweitern wollen. Für sie finden sich entsprechende Installations- und Anwendungshinweise im Anhang des Buches.

Als zusätzliche Bereicherung enthält die CD den *Position und Proper Motion* Katalog von S. Röser und U. Bastian (verlegt im Spektrum Akademie Verlag) mit detaillierten Informationen zu rund 470 000 Sternen sowie die *Asteroid Orbital Elements* Datenbank von E. Bowell mit Bahninformationen von über 50 000 Kleinenplaneten. Die entsprechenden Dateien wurden freundlicherweise vom Astronomischen Rechen-Institut Heidelberg und dem Lowell Observatorium zur Verfügung gestellt, bei denen wir uns – ebenso wie bei den Autoren – an dieser Stelle ausdrücklich bedanken. Zum leichteren Gebrauch der recht umfangreichen Kataloge werden auf der CD verschiedene Zusatzprogramme bereitgestellt, mit deren Hilfe die Daten für den Einsatz in den Programmen **FOTO** und **NUMINT** aufbereitet werden können.

Hinweisen möchten wir die Leser an dieser Stelle auch auf die neu eingerichtete Webseite <http://www.springer.de/phys-de/books/astropc/>, auf der wir zusammen mit dem Verlag nützliche Informationen aus dem Umfeld des Buches anbieten. So findet man hier zum Beispiel die Quelltexte der Pascal-Programme aus der zweiten Auflage von „Astronomie mit dem Personal Computer“ und Verweise auf nützliche Internet-Ressourcen. Bei Bedarf werden hier auch Korrekturen und Software-Updates zur Verfügung gestellt.

Herrn Dr. Ch. Caron und Frau Reichel-Mayer vom Springer-Verlag in Heidelberg gilt unser besonderer Dank für die gute Zusammenarbeit und Unterstützung bei der Herstellung dieses Buches. Ebenso möchten wir uns bei Dr. L. Weidinger bedanken, dessen Unterstützung die Portierung unserer Programme auf eine Linux-Umgebung erst möglich gemacht hat.

Vorwort zur zweiten Auflage

Seit dem Erscheinen von „*Astronomie mit dem Personal Computer*“ haben uns zahlreiche Anregungen und Hinweise erreicht. Dies und das Interesse des Verlages an einer Neuauflage waren uns Anlaß, das Buch zu überarbeiten und umfangreiche inhaltliche Ergänzungen vorzunehmen.

Als erste wichtige Neuerung ist ein Kapitel über die Störungsrechnung hinzugekommen. Es zeigt, wie sich die gravitativen Störungen der großen Planeten bei der Ephemeridenrechnung für Kleinplaneten und Kometen berücksichtigen lassen. Das vorgestellte Programm NUMINT ermöglicht die Berechnung genauer Positionen, die zum Aufsuchen der oft sehr lichtschwachen Objekte am Himmel eine wichtige Hilfe sind. Das jetzt verfügbare Werkzeug der Störungsrechnung rundet die Thematik der Kapitel über Ephemeridenrechnung, Bahnbestimmung und Astrometrie ab. Das zweite neu hinzugekommene Kapitel befaßt sich mit der Berechnung von physischen Ephemeriden der großen Planeten und der Sonne und schließt damit eine inhaltliche Lücke. Der Sternfreund verfügt nun über die Hilfsmittel zur Vorbereitung und Auswertung interessanter Planetenbeobachtungen. Weitere Ergänzungen betreffen hauptsächlich die Auf- und Untergangsrechnung für die Planeten und die Berechnung der lokalen Umstände von Sonnenfinsternissen.

Schließlich gibt es nun nur noch eine Version der Programmdiskette, die dem Buch direkt beigefügt ist. Nachdem die Firma Application Systems Heidelberg inzwischen den zu Turbo Pascal von Borland kompatiblen Compiler Pure Pascal für Atari ST/TT Rechner auf den Markt gebracht hat, erübrigert sich eine spezielle Atari-Version der Programmdiskette. Die beigelegte Diskette kann ohne Änderungen sowohl mit Turbo Pascal auf IBM-kompatiblen PCs als auch mit Pure Pascal auf Atari-Computern verwendet werden. Entsprechende Installationshinweise sind der Datei **AAREADME.DOC** auf der Programmdiskette zu entnehmen.

Wir danken dem Springer-Verlag für die gute Zusammenarbeit und der Firma Application Systems Heidelberg für ihre Unterstützung.

September 1993

O. Montenbruck und T. Pfleger

Vorwort zur ersten Auflage

Wer sich heute privat oder beruflich mit der Berechnung astronomischer Ereignisse befaßt, wird dafür zwangsläufig auf einen Computer zurückgreifen. Dies gilt insbesondere, seit sich der „Personal Computer“ als allgegenwärtiger Gehilfe einen festen Platz in unserem Leben erobert hat. Rechenleistungen, die vor kurzem noch nicht denkbar gewesen wären, stehen nun einem breiten Kreis von Anwendern direkt am Schreibtisch zur Verfügung.

Im gleichen Maß wie die technischen Möglichkeiten des Computers ist aber auch der Bedarf an leistungsfähigen, das heißt schnellen und genauen Programmen gewachsen. So erscheint der Wunsch, auf herkömmliche astronomische Jahrbücher so weit wie möglich zu verzichten, heute als völlig selbstverständlich.

Wir haben deshalb gerne die Anregung des Verlags aufgegriffen, in dem nun vorliegenden Buch die Umsetzung grundlegender Aufgabenstellungen der sphärischen Astronomie, der Ephemeridenrechnung und der Himmelsmechanik in Computerprogramme darzustellen.

Dem Leser, der selbst Programme entwickelt, bietet „*Astronomie mit dem Personal Computer*“ eine umfangreiche Bibliothek von Pascal-Unterprogrammen mit fertigen Lösungen zu einer Vielzahl immer wiederkehrender Teilprobleme. Hierzu zählen Routinen für alle gängigen Koordinatentransformationen und zur Zeit- und Kalenderrechnung ebenso wie zur Behandlung des Zweikörperproblems. Gesonderte Unterprogramme ermöglichen die genaue Berechnung der Positionen von Sonne, Mond und Planeten unter Berücksichtigung der gegenseitigen Störungen. Dank der weiten Verbreitung der Sprache Pascal und des Verzichts auf rechner-abhängige Befehle können die Programme auf allen gängigen Computern vom PC bis hin zum Großrechner eingesetzt werden. Die Vielzahl der vorgestellten Routinen trägt hoffentlich ein wenig dazu bei, daß nicht jeder Anwender „das Rad neu erfinden“ muß und sich statt dessen auf seine eigentliche Aufgabe konzentrieren kann.

Jedes Kapitel des Buches behandelt ein möglichst abgegrenztes Thema und schließt mit einem vollständigen Hauptprogramm. Ausgehend von einfachen Fragestellungen wie der Bestimmung von Auf- und Untergangszeiten oder der Berechnung von Mond- und Planetenephemeriden, werden weiterführende Themen bis hin zur Berechnung von Sonnenfinsternissen und Sternbedeckungen behandelt. Die Programme zur astrometrischen Auswertung von Sternfeldaufnahmen und zur Bahnbestimmung ermöglichen es, selbst Bahnelemente von Kometen oder Planetoiden zu bestimmen. Auch Lesern ohne Programmiererfahrung stehen damit sofort fertige Anwendungen zur Verfügung.

Die astronomischen und numerischen Grundlagen, die zur Lösung eines speziellen Problems benötigt werden, sind jeweils so ausführlich dargestellt, wie es für ein Verständnis der vorgestellten Programme nötig ist. Mit diesem Wissen ist es dann auch möglich, alle Programme individuellen Wünschen anzupassen. Die enge Verbindung von Theorie und Praxis bietet darüber hinaus die Möglichkeit, die teilweise komplizierten Zusammenhänge leichter nachzuvollziehen, als es die Darstellung in klassischen Lehrbüchern erlaubt. Zusammenfassend hoffen wir, dem Leser mit diesem Buch eine fundierte Grundlage für den Einsatz des Computers in der Astronomie in die Hand zu geben.

An dieser Stelle möchten wir uns bei Dr. H. U. Daniel und R. Michels vom Verlag für die angenehme Zusammenarbeit und das Interesse an der Herausgabe dieses Buches bedanken. Unser Dank gilt ferner allen unseren Freunden und Kollegen, die mit ihren Ideen und Anregungen sowie durch ihre Unterstützung bei der Korrektur des Manuskriptes und beim Testen der Programme einen wesentlichen Anteil am Gelingen des Werkes hatten.

August 1989

O. Montenbruck und T. Pfleger

Inhalt

1 Einführung	1
1.1	Anwendungsbeispiele	1
1.2	Astronomie und Numerik	2
1.3	Programmiersprache und -technik	3
2 Koordinatensysteme	7
2.1	Aller Anfang ist schwer	7
2.2	Kalender und julianisches Datum	14
2.3	Ekliptik und Äquator	17
2.4	Präzession	20
2.5	Geozentrische Koordinaten und die Sonnenbahn	24
2.6	Das Programm COCO	27
3 Auf- und Untergangsrechnung	35
3.1	Das Horizontsystem des Beobachters	35
3.2	Sonne und Mond	38
3.3	Sternzeit und Stundenwinkel	39
3.4	Weltzeit und Ephemeridenzeit	41
3.5	Parallaxe und Refraktion	44
3.6	Auf- und Untergänge	47
3.7	Quadratische Interpolation	48
3.8	Das Programm SUNSET	50
3.9	Das Programm PLANRISE	57
4 Kometenbahnen	59
4.1	Form und Lage der Bahn	59
4.2	Der Ort in der Bahn	61
4.3	Die numerische Behandlung der Keplergleichung	65
4.4	Parabelnahe Bahnen	68
4.5	Die Gaußschen Vektoren	72
4.6	Die Lichtlaufzeit	76
4.7	Das Programm COMET	77
5 Störungsrechnung	85
5.1	Bewegungsgleichung	86
5.2	Planetenkoordinaten	89
5.3	Numerische Integration	91

5.4	Oskulierende Bahnelemente	97
5.5	Das Programm NUMINT	100
5.6	Die Asteroid Orbital Elements Datenbank	108
6	Planetenbahnen	111
6.1	Reihenentwicklung des Keplerproblems	112
6.2	Störungsterme	115
6.3	Numerische Behandlung der Reihenentwicklungen	118
6.4	Scheinbare und astrometrische Koordinaten	124
6.4.1	Aberration und Lichtlaufzeit	124
6.4.2	Die Nutation	127
6.5	Das Programm PLANPOS	129
7	Physische Planetenephemeriden	135
7.1	Rotation	135
7.1.1	Der Positionswinkel der Achse	136
7.1.2	Planetographische Koordinaten	139
7.2	Beleuchtungsverhältnisse	146
7.2.1	Phase und Elongation	146
7.2.2	Der Positionswinkel der Sonne	147
7.2.3	Scheinbare Helligkeit	148
7.2.4	Scheinbarer Durchmesser	150
7.3	Das Programm PHYS	150
8	Die Mondbahn	155
8.1	Allgemeine Beschreibung der Mondbahn	155
8.2	Die Brownsche Mondtheorie	159
8.3	Tschebyscheff-Approximation	168
8.4	Das Programm LUNA	174
9	Sonnenfinsternisse	179
9.1	Mondphasen und Finsternisse	179
9.2	Die Geometrie der Finsternis	181
9.3	Geographische Koordinaten und die Abplattung der Erde	186
9.4	Die Dauer der Finsternis	189
9.5	Sonnen- und Mondkoordinaten	190
9.6	Das Programm ECLIPSE	192
9.7	Die lokalen Umstände einer Sonnenfinsternis	200
9.8	Das Programm ECLTIMER	203
10	Sternbedeckungen	205
10.1	Scheinbare Sternkoordinaten	206
10.2	Die geozentrische Konjunktion	210
10.3	Die Fundamentalebene	214
10.4	Ein- und Austritt	216
10.5	Das Programm OCCULT	219
10.6	Abschätzung von $\Delta T = ET - UT$ aus Beobachtungen	228

	Inhalt	XIII
11 Bahnbestimmung	231	
11.1 Die Festlegung der Bahn durch zwei Ortsvektoren	231	
11.1.1 Das Sektor-zu-Dreieck-Verhältnis	232	
11.1.2 Die Bahnelemente	235	
11.2 Das verkürzte Gauß-Verfahren	239	
11.2.1 Geometrie der geozentrischen Beobachtungen	239	
11.2.2 Iteration der Dreiecksflächenverhältnisse	242	
11.2.3 Mehrfache Lösungen	243	
11.3 Die vollständige Gauß-Methode	244	
11.3.1 Die Gauß-Lagrangesche Gleichung	244	
11.3.2 Verbesserte Iteration der Dreiecksflächenverhältnisse	246	
11.3.3 Lichtlaufzeit	247	
11.4 Das Programm GAUSS	249	
12 Astrometrie	259	
12.1 Die fotografische Abbildung	259	
12.2 Die Plattenkonstanten	262	
12.3 Ausgleichsrechnung	264	
12.4 Das Programm FOTO	267	
12.5 Der Position and Proper Motion Sternkatalog	272	
Anhang	275	
A.1 Die beigelegte CD	275	
A.1.1 Inhalt	275	
A.1.2 Systemvoraussetzungen	276	
A.1.3 Ausführung der Programme	277	
A.2 Übersetzen und Binden der Programme	279	
A.2.1 Allgemeine Hinweise zur Rechneranpassung	279	
A.2.2 Microsoft Visual C++ für Windows 95/98/NT	280	
A.2.3 GNU C++ für Linux	281	
A.3 Verzeichnis der Bibliotheksfunktionen	283	
Bezeichnungen	289	
Glossar	293	
Literaturverzeichnis	297	
Sachverzeichnis	305	

1. Einführung

Die letzten Jahre brachten eine kontinuierliche Steigerung der Leistungsfähigkeit kleiner Computer bei gleichzeitigem Sinken ihrer Preise. Demzufolge stehen heute vielen an der Astronomie Interessierten solche Geräte zur Verfügung. Der Wunsch, den Computer auch für astronomische Berechnungen heranzuziehen, ist naheliegend. Welche konkreten Vorteile können sich durch den Einsatz des eigenen Rechners ergeben, zumal man die wichtigsten Daten, die zur Beobachtung benötigt werden, doch ebensogut in einem der zahlreichen Jahrbücher nachschlagen kann?

1.1 Anwendungsbeispiele

Betrachten wir zunächst einmal die Berechnung der Auf- und Untergangszeiten von Sonne und Mond. Von einem Beobachtungsort zum anderen ergeben sich schnell Zeitdifferenzen, die man nicht mehr vernachlässigen will. In einem Jahrbuch werden die Auf- und Untergangszeiten aber in der Regel nur für einige wenige Orte angegeben. Man ist deshalb häufig auf eine Interpolation oder die Anwendung von Nomogrammen angewiesen, wenn man die Zeiten für einen beliebigen Ort bestimmen möchte. Hier ist es wesentlich komfortabler und genauer, wenn der Rechner ohne zusätzliche Überlegungen direkt die gesuchten Daten liefern kann.

Den umfangreichsten Teil eines Jahrbuchs bilden die seitenlangen Tafeln der Sonnen-, Mond- und Planetenpositionen. Von den dort abgedruckten Daten ist oft nur ein kleiner Teil von Interesse. Mit Hilfe von entsprechenden Programmen lassen sich die tatsächlich benötigten Werte schnell berechnen und gleichzeitig in jedem gewünschten Koordinatensystem darstellen.

Noch wichtiger ist die Möglichkeit, die Bahn eines Himmelskörpers per Programm selbst zu berechnen, im Fall von Kometen. Bei Neuentdeckungen sind aktuelle Ephemeriden oft gar nicht oder nur mit Verzögerungen erhältlich. Ein einziger Satz von Bahnelementen genügt dagegen schon, um die Bewegung des Kometen ausreichend genau verfolgen zu können.

Sonnenfinsternisse sind eindrucksvolle Himmelsschauspiele, die Amateurastronomen zu teils weiten Reisen anregen. Obwohl offizielle Quellen zuverlässige Daten geben, ist es wünschenswert, schon vor deren Verfügbarkeit eigene Planungen anstellen zu können. Falls man Angaben für einen Beobachtungsort benötigt, der nicht in den Jahrbüchern berücksichtigt wurde, hilft hier nur ein eigenes Programm zur Ermittlung der lokalen Umstände weiter.

Wer Sternbedeckungen durch den Mond beobachten will, benötigt dazu Kontaktzeiten für seinen Beobachtungsort. Soweit Jahrbücher solche Vorhersagen überhaupt enthalten, beschränken sich diese in der Regel auf einige ausgewählte Groß-

städte. Wer an einem anderen Ort beobachtet, ist somit wieder auf Abschätzungen angewiesen. Das in diesem Buch vorgestellte Programm *Occult* erlaubt die Suche und Berechnung von Sternbedeckungen für eine beliebige Auswahl von Sternen.

Viele Amateurastronomen widmen sich der Astrofotografie. Mit Hilfe von Vergleichssternen bekannter Koordinaten läßt sich die Position eines Planetoiden oder Kometen aus selbstgemachten Aufnahmen bestimmen. Verfügt man über mindestens drei Aufnahmen, die in einem Abstand voneinander aufgenommen wurden, dann lassen sich im Rahmen einer Bahnbestimmung die Bahnelemente berechnen. Wir stellen mit den Programmen *Foto*, *Gauss* und *Comet* das notwendige Handwerkszeug für die recht aufwendigen Rechnungen bereit. Planetoidenbeobachter werden die Möglichkeit schätzen, mit dem Programm *Numint* genaue Ephemeriden unter Berücksichtigung der gravitativen Störungen der großen Planeten erhalten zu können – auch wenn die verfügbaren Bahnelemente sich auf eine schon mehrere Jahre zurückliegende Epoche beziehen.

1.2 Astronomie und Numerik

Eine Einführung in die Berechnung astronomischer Phänomene mit Computerprogrammen wäre unvollständig ohne eine Diskussion der dabei erreichbaren Genauigkeit. Wesentlich ist dabei zunächst die Feststellung, daß die Qualität solcher Programme vor allem von der mathematischen und physikalischen Beschreibung des jeweiligen Problems sowie der Güte der verwendeten Algorithmen abhängt. Dagegen erweist sich vielfach die Annahme als Irrtum, daß allein die Verwendung einer Rechnerarithmetik mit doppelter Stellenzahl die Korrektheit eines bestimmten Programms garantieren würde. Auch ein exakt gerechnetes Programm muß ungenaue Resultate liefern, wenn der verwendete Rechenweg keine bessere Genauigkeit erlaubt.

Ein Beispiel hierfür ist die Berechnung von Ephemeriden nach den Gesetzen des Zweikörperproblems. Dabei wird die Bahn eines Planeten in einfacher Weise durch eine Ellipse dargestellt. Neben der Anziehungskraft der Sonne, die für diese Bahnform verantwortlich ist, wirken jedoch weitere Kräfte von seiten der übrigen Planeten. Sie führen zu periodischen und langfristigen Störungen, die bei den inneren Planeten etwa eine Bogenminute, bei den äußeren Planeten sogar bis zu ein Grad betragen können. Die Theorie der Keplerbahnen stellt also lediglich eine Näherung der tatsächlichen Verhältnisse dar. Fordert man eine höhere Genauigkeit, dann ist es unumgänglich, kompliziertere Beschreibungen der Planetenbahnen heranzuziehen, die auch die gegenseitigen Störungen der Planeten modellieren. Mit der Verwendung einer hochgenauen Rechenarithmetik wäre noch kein entscheidender Schritt getan!

Wir haben bei der Entwicklung unserer Programme eine Genauigkeit angestrebt, die etwa im Bereich gängiger astronomischer Jahrbücher liegt. Die Fehler der grundlegenden Routinen zur Bestimmung der Sonnen-, Mond- und Planetenkoordinaten betragen rund $1''\text{--}3''$. Diese Genauigkeit reicht beispielsweise zur Berechnung von Sonnenfinsternissen oder Sternbedeckungen aus und sollte deshalb auch für die meisten anderen Anwendungen genügen.

Wer konsequent genau rechnen will, muß sich jedoch zunächst über die exakten Definitionen der verwendeten Koordinatensysteme im klaren sein. Leider zeigt die Praxis, daß hier gerne Angaben in unterschiedlichen Systemen miteinander verglichen werden. Schnell hört man dann, daß ein Programm ungenau rechne, weil in einem Jahrbuch andere Zahlenwerte abgedruckt sind. Oft läßt sich die Diskrepanz aber allein durch die unterschiedliche Bedeutung der Koordinatenangaben erklären, die der Benutzer nicht erkannt hat und deshalb einen „Fehler“ vermutet. Wir haben deshalb an allen relevanten Stellen auf wichtige Korrekturen wie Präzession und Nutation oder Aberration und Lichtlaufzeit hingewiesen. Auch auf den Unterschied zwischen Weltzeit und Ephemeridenzeit, der immer wieder Schwierigkeiten bereitet, wird mehrfach eingegangen. Der Leser sollte sich durch die häufigen Hinweise auf die genannten Effekte aber nicht verunsichern lassen. Wir hoffen vielmehr, daß diese Form der Darstellung und die direkte Umsetzung in Programme ein besseres Gefühl für die Größenordnung und die praktische Bedeutung der einzelnen Korrekturen ermöglichen. Die gelegentlichen Wiederholungen schienen uns nötig, um jedes Kapitel – und damit die Darstellung eines eng umrissenen Themas – auch aus dem Zusammenhang heraus lesen zu können. Da aber die Programme häufig aufeinander aufbauen, sollte man bei Gelegenheit auch die anderen Kapitel der Reihe nach lesen.

Die Leistungsfähigkeit eines Programms hängt über die mathematische Beschreibung hinaus auch von der Verwendung geeigneter Rechenverfahren ab. Sie werden jeweils im Zusammenhang mit den verschiedenen Anwendungen besprochen. Ein Beispiel hierfür ist die Auswertung von Winkelfunktionen bei der Berechnung der periodischen Bahnstörungen des Mondes und der Planeten. Durch Ausnutzung der Additionstheoreme und Rekursionsbeziehungen für die trigonometrischen Funktionen lassen sich die Rechenzeiten der Programme *Planpos* und *Luna* deutlich verkürzen. Benötigt man für bestimmte Aufgaben besonders viele Positionen eines Himmelskörpers in kurzen Abständen, dann lohnt es sich, diese zunächst durch Tschebyscheff-Polynome zu approximieren. Man erhält so ohne Genauigkeitsverlust für einen begrenzten Zeitraum gültige Ausdrücke für die Koordinaten, die besonders schnell und einfach ausgewertet werden können. Hiervon wird in den Programmen *Eclipse*, *EclTimer* und *Occult* Gebrauch gemacht. Weitere Aspekte der numerischen Mathematik werden bei der Auflösung von Gleichungen mittels Newton-Verfahren, Regula Falsi oder quadratischer Interpolation angesprochen. Im Rahmen der astrometrischen Auswertung von Fotoplatten wird darüberhinaus ein einfacher, aber sicherer Algorithmus zur Ausgleichsrechnung vorgestellt.

1.3 Programmiersprache und -technik

Neben der stürmischen Entwicklung der Hardware hat sich in den letzten Jahren auch die Softwaretechnik stark weiterentwickelt. Die immer leistungsfähigeren und umfangreicheren Anwendungsprogramme sind in der Regel mit einer grafischen Benutzeroberfläche ausgestattet, wobei sich Microsoft Windows als Standard etablieren konnte. Aber auch aus der Sicht des Programmierers hat sich viel

verändert: mit dem Wunsch oder der Notwendigkeit, solche grafisch ausgefeilten und ereignisorientierten, auf Aktionen des Benutzers reagierende Anwendungen zu erstellen, haben sich objektorientierte Methoden der Softwareentwicklung in der Praxis etabliert. Die Datenverarbeitung in den Unternehmen hat darüberhinaus neue Trends in der Entwicklung ausgelöst: Software soll netzwerkfähig und weitgehend unabhängig von Hardwareplattform und Betriebssystem eingesetzt und entwickelt werden können. Ergebnis dieser Entwicklung ist die Verbreitung objektorientierter Programmiersprachen wie C++ und neuerdings das C++ sehr ähnliche JAVA, das seine Vorteile vor allem im Bereich der Datenverarbeitung in Firmennetzen und dem Internet ausspielen kann.

Im Bereich der technisch-wissenschaftlichen Anwendungsentwicklung werden nach wie vor strukturierte Programmiersprachen wie FORTRAN und ADA eingesetzt, die jedoch eine deutliche Modernisierung erfahren haben (FORTRAN 90, ADA 95). Darüber hinaus gewinnt das objektorientierte C++ aber gerade in diesem Bereich neue Anhänger, was vor allem auf seine universelle Verwendbarkeit, die gute Leistung der übersetzten Programme und die allgemeine Verfügbarkeit zurückzuführen sein dürfte. C++ erlaubt sowohl das strukturierte als auch das objektorientierte Programmieren, und wird deshalb oft als hybride Sprache bezeichnet. Zwar können astronomische und numerische Algorithmen die Vorteile objektorientierter Programmierung naturgemäß nur eingeschränkt nutzen. Sollen solche Kernelalgorithmen jedoch in Programmen mit grafischer Benutzeroberfläche eingesetzt werden, dann erweist sich C++ als flexible und mächtige Sprache, die Algorithmik, Benutzerführung und Grafik gleichermaßen effizient zu implementieren gestattet. Dies begründet unsere Entscheidung für C++ als Programmiersprache für die vorliegende Neuauflage dieses Buches.

Es sei an dieser Stelle aber nicht verschwiegen, daß C++ auch manche Defizite gegenüber anderen Sprachen aufweist, die die Erstellung komplexer naturwissenschaftlicher Programme erschweren. Hierzu zählen etwa die schlechte Unterstützung ein- und mehrdimensionaler Felder, das Fehlen lokaler Prozeduren, das mangelhafte Modulkonzept sowie fehlende Indexprüfungen und freizügige Typumwandlungen.

Entsprechend den behandelten Problemstellungen und den zur Verfügung stehenden sprachlichen Möglichkeiten, kommen in den hier vorgestellten Programmen Elemente der strukturierten und der objektorientierten Programmierung gleichberechtigt nebeneinander zum Tragen. Wie in der früheren Pascal Implementierung sind die wesentlichen Ziele dabei eine transparente und verständliche Umsetzung der zugrundeliegenden Algorithmen sowie die Schaffung modularer und wieder verwendbarer Komponenten.

Gerade bei der Erstellung astronomischer Programme begegnet man immer wieder bestimmten grundlegenden Teilproblemen, die sich dafür besonders gut eignen. Beispiele hierfür sind die Auswertung mathematischer Funktionen, die Umrechnung von Koordinatenangaben in verschiedenen Systeme oder die Bestimmung genauer Positionen der Planeten, der Sonne oder des Mondes. Es wäre sehr mühsam, sich jedesmal von neuem mit diesen Aufgaben befassen zu müssen. Hat man die immer wieder benötigten Funktionen in Form einer Bibliothek le-

stungsfähiger und zuverlässiger Unterprogramme zur Verfügung, dann kann man sich auf die eigentliche Funktionalität des neu zu erstellenden Programms konzentrieren und kommt damit schneller und leichter ans Ziel.

Viele der Operatoren, Funktionen und Klassen, die wir in diesem Buch vorstellen, bilden so den Grundstock einer leistungsfähigen astronomischen Programm-bibliothek. Sie unterteilt sich in Module für

- Hilfsfunktionen zur Ein- und Ausgabe (`APC_IO.cpp`),
- mathematische und astronomische Konstanten (`APC_Const.cpp`),
- technisch-wissenschaftliche Klassen und Funktionen
(`APC_Math.cpp`, `APC_VecMat3D.cpp`),
- Klassen und Funktionen zu Problemen der numerischen Mathematik
(`APC_Cheb.cpp`, `APC_DE.cpp`),
- Funktionen und Hilfsklassen zur Zeit- und Kalenderrechnung
(`APC_Time.cpp`),
- diverse Funktionen zur sphärischen Astronomie (`APC_Spheric.cpp`),
- spezielle Funktionen zur Präzession und Nutation (`APC_PrecNut.cpp`),
- Funktionen zur Berechnung elliptischer, parabolischer und hyperbolischer Keplerbahnen (`APC_Kepler.cpp`),
- Funktionen zur Berechnung genauer Positionen von Sonne, Mond und Planeten unter Berücksichtigung der verschiedenen Bahnstörungen (`APC_Sun.cpp`, `APC_Moon.cpp`, `APC_Planets.cpp`) sowie
- Funktionen zur Berechnung physischer Planetenephemeriden
(`APC_Phys.cpp`).

Die einzelnen Komponenten der Bibliothek werden schrittweise in den entsprechenden Kapiteln dieses Buches entwickelt und erläutert. Mit Hilfe der zugehörigen Headerdateien (`APC_*.h`) können die jeweils benötigten Module in den Anwendungsprogrammen auf einfache Weise bekannt gemacht werden.

Bei der Verwendung von Klassen und Objekten haben wir uns auf ausgewählte Fälle beschränkt, da ein übereifriger Einsatz dieser Sprachelemente die Programme auch umständlicher und schlechter lesbar machen kann.

Unmittelbare Vorteile ergeben sich zum Beispiel aus der Definition eigener Klassen `Vec3D` und `Mat3D` für Vektoren und Matrizen in Verbindung mit der Möglichkeit des Überladens von Operatoren. C++ erlaubt es, die Operatoren der Sprache (wie z.B. `+`, `-` oder `*`) so zu erweitern, daß sie auch mit benutzerdefinierten Datentypen umgehen können. Anstelle umständlicher Funktionsaufrufe kann die Addition von Vektoren oder die Multiplikation von Matrizen wesentlich eleganter mit Hilfe dieser Operatoren durchgeführt werden. Die Verwendung geeigneter definierter Klassen erlaubt somit eine weitgehende Entsprechung von Anweisungen und Gleichungen und trägt so zur besseren Übersichtlichkeit und Lesbarkeit der Quelltexte bei.

Weiterhin schien es uns sinnvoll, Hilfsklassen `Time`, `DateTime` und `Angle` zur Vereinfachung einer formatierten Ausgabe von Zeitpunkten und Winkeln zu defi-

nieren. Sie unterstützen die Verwendung des Schiebeoperators (`<<`) und erlauben so eine einfache und C++ konforme Schreibweise dieser Ausgabeoperationen.

Beispiele für einen gewinnbringenden Einsatz objektorientierter Techniken bei der Implementierung numerischer Verfahren sind die Klasse `SolverDE` zur numerischen Integration von Differentialgleichungen und die Klasse `SolverLSQ` zur Lösung linearer Ausgleichsprobleme. Während sich die elementaren Rechenschritte naturgemäß nicht von der Realisierung in klassischen Programmiersprachen unterscheiden, erlaubt das Klassenkonzept eine weitgehende Abstraktion der bereitgestellten Funktionen und eine optimale Kapselung der benötigten Daten. Aufgaben wie die Bereitstellung von Arbeitsfeldern liegen nicht mehr in der Verantwortung des Benutzers, sondern werden selbstständig vom Konstruktor entsprechender Objekte durchgeführt.

2. Koordinatensysteme

In der Astronomie werden nebeneinander eine ganze Reihe verschiedener Koordinatensysteme verwendet, um die Positionen von Sternen und Planeten festzulegen. Man unterscheidet hierzu zwischen heliozentrischen Koordinaten, die sich auf die Sonne als Ursprung beziehen, und geozentrischen Koordinaten, die vom Erdmittelpunkt aus gezählt werden. Ekliptikale Koordinaten geben den Ort eines Punktes bezüglich der Erdbahnebene an, während äquatoriale Koordinaten relativ zur Lage des Erd- bzw. Himmelsäquators gemessen werden. Die langsame Verschiebung dieser Bezugsebenen durch die Präzession macht zusätzlich eine Unterscheidung nach dem Äquinoktium des verwendeten Koordinatensystems nötig. Obwohl es eigentlich genügen würde, sich ein für alle Mal auf eine feste Zählweise der Koordinaten zu einigen, hat jedes System Eigenschaften, die es für bestimmte Zwecke besonders geeignet machen.

Mit der Vielfalt möglicher Koordinatensysteme stellt sich allerdings auch die oft lästige Aufgabe, Koordinaten von einem System in ein anderes zu übertragen. Das Programm Coco („Koordinaten-Konvertierung“) soll dabei behilflich sein. Es erlaubt die genaue Umrechnung zwischen ekliptikalnen und äquatorialen sowie zwischen geozentrischen und heliozentrischen Koordinaten verschiedener Äquinoktien. Die einzelnen Transformationen, die im folgenden beschrieben werden, sind jeweils als eigene Funktionen formuliert. Sie können deshalb auch an anderen Stellen verwendet werden und bilden eine wichtige Grundlage für die späteren Programme.

2.1 Aller Anfang ist schwer

Die Programmiersprache C++ verfügt über mächtige Standard-Bibliotheken, in denen bereits viele mathematische Funktionen enthalten sind. Dennoch erweist es sich für die weitere Arbeit als hilfreich, zu Beginn selbst einige grundlegende Bibliotheksmodule zu definieren. Hierzu gehört zunächst die Definition wichtiger Konstanten, auf die später immer wieder zurückgegriffen wird. Das entsprechende Modul `APC_Const` besteht lediglich aus einer Schnittstellendatei `APC_Const.h`, die über eine `#include` Anweisung in andere Module aufgenommen werden kann. Hierin sind neben mathematischen Konstanten wie der Zahl π und der Faktoren zur gegenseitigen Umrechnung von Grad- und Bogenmaß auch astronomische und physikalische Konstanten wie die Länge der Astronomischen Einheit und der Wert der Lichtgeschwindigkeit enthalten.

```
const double pi      = 3.14159265358979324;
const double pi2     = 2.0*pi;
```

```

const double Rad      = pi / 180.0;
const double Deg      = 180.0 / pi;
const double Arcs     = 3600.0*180.0/pi;
...
const double AU        = 149597870.0;    // [km]
const double c_light   = 173.14;         // [AE/d]

```

Das Modul APC_Math stellt unter anderem die Funktionen

```

//-----
// Frac: Liefert den Nachkommateil einer Zahl
//-----
double Frac ( double x )
{
    return x-floor(x);
}

```

und

```

//-----
// Modulo: Berechnet x mod y
//-----
double Modulo ( double x, double y )
{
    return y*Frac(x/y);
}

```

zur Verfügung, mit denen sich der gebrochene Anteil und der Teilungsrest einer Zahl berechnen lassen. Die Funktionen werden beispielsweise immer dann benötigt, wenn man die monoton wachsende mittlere Länge eines Himmelskörpers auf den Bereich von 0° bis 360° abbilden will.

Darüber hinaus enthält APC_Math zwei Funktionen Ddd und DMS, die eine sexagesimale Darstellung von Zeiten und Winkeln unterstützen. Bei der Angabe von Winkeln im Gradmaß werden Bruchteile eines Grades normalerweise in Bogenminuten und Bogensekunden angegeben. Dies entspricht der bei Zeiten üblichen Darstellung in Minuten und Sekunden. Eine Umwandlung der beiden Darstellungen ist normalerweise höchstens einmal bei der Ein- oder Ausgabe einer Größe notwendig. Solange alle Werte positiv sind, bereitet dies prinzipiell keine Schwierigkeiten. Man muß allerdings einige Vorkehrungen treffen, um auch bei negativen Winkeln das Vorzeichen immer korrekt zu berücksichtigen. Die Wirkung der Funktionen Ddd und DMS wird am einfachsten an einigen Beispielen deutlich:

Dd	D	M	S
15.50000	15	30	00.0
-8.15278	-8	09	10.0
0.01667	0	1	0.0
-0.08334	0	-5	0.0

Bei der Umwandlung einer negativen Zahl Dd in Grad, Minuten und Sekunden wird jeweils nur die führende der drei Zahlen D, M und S negativ. Zu beachten ist noch, daß in beiden Funktionen D und M, die naturgemäß nur ganzzahlige

Werte annehmen können, als int-Größen definiert sind, während S als double-Zahl vereinbart ist.

```
-----
// Ddd: Umrechnung eines in Grad, Bogenminuten und Bogensekunden gegebenen
//       Winkels in dezimale Darstellung
//   D,M,S      Winkelgrade, Bogenminuten, Bogensekunden
//   <return>    Winkel in dezimaler Darstellung
-----
double Ddd( int D, int M, double S )
{
    double sign;
    if ( (D<0) || (M<0) || (S<0) ) sign = -1.0; else sign = 1.0;
    return sign * ( fabs(D)+fabs(M)/60.0+fabs(S)/3600.0 );
}

-----
// DMS: Ermittelt Grad, Bogenminuten und Bogensekunden zu gegebenem Winkel
//   Dd          Winkel in Grad in dezimaler Darstellung
//   D,M,S      Winkelgrade, Bogenminuten, Bogensekunden
// -----
void DMS ( double Dd, int& D, int& M, double& S )
{
    double x;
    x = fabs(Dd);  D = int(x);
    x = (x-D)*60.0; M = int(x);  S = (x-M)*60.0;
    if (Dd<0.0) { if (D!=0) D*=-1; else if (M!=0) M*=-1; else S*=-1.0; }
}
```

Da die Umrechnung von Winkeln in Grad, Minuten und Sekunden nahezu ausschließlich bei der Ausgabe benötigt wird, lohnt es sich, beide Funktionen geeignet zusammenzufassen. Wir machen dabei erstmals von der in C++ gegebenen Möglichkeit Gebrauch, eine eigene Klasse zu definieren:

```
// Formatangabe fuer Ausgaben von Winkeln mit dem Operator << der Klasse Angle
enum AngleFormat {
    Dd,      // dezimale Darstellung
    DMM,     // Grade und ganze Bogenminuten
    DMMm,    // Grade und Bogenminuten in dezimaler Darstellung
    DMMSS,   // Grade, Bogenminuten und ganze Bogensekunden
    DMMSSs // Grade, Bogenminuten und Bogensekunden in dezimaler Darstellung
};

//
// Hilfsklasse zur Ausgabe von Winkeln
//
class Angle
{
public:
    // Konstruktor
    Angle (double alpha, AngleFormat Format=Dd);
    // Zugriffsfunktionen
    void Set (AngleFormat Format=Dd);
    // Ausgabeoperator fuer Winkel
    friend ostream& operator << (ostream& os, const Angle& alpha);
```

```

private:
    double      m_angle;
    AngleFormat m_Format;
};
```

Objekte der Klasse `Angle` können über den gleichnamigen Konstruktor durch Angabe des Winkels und eines Formatbezeichners erzeugt werden. Die entsprechenden Angaben werden in Attributen der Klasse gespeichert und bei der späteren Ausgabe mittels des Schiebeoperators `<<` verwendet. Dabei wird zunächst nach Bedarf die Zerlegung des Winkels in Grad, Minuten und Sekunden durchgeführt. Anschließend werden die resultierenden Größen einzeln formatiert ausgegeben. Die Definition des Schiebeoperators unterstützt die gängigen Manipulatoren für Stellenzahl, Vorzeichen, etc., so daß die Ausgabe in weiten Grenzen frei gestaltet werden kann. Die Implementierung der Klasse ist angesichts der Fallunterscheidungen für die einzelnen Formate recht umfangreich und deshalb an dieser Stelle nicht im einzelnen wiedergegeben. Dagegen gestaltet sich die Anwendung sehr einfach, wie das folgende kleine Programm zeigt:

```

#include <iostream>
#include <iomanip>
#include "APC_Math.h"
void main()
{
    using namespace std;
    double x=12.3456;
    cout << setprecision(2) << setw(12) << Angle(x,Dd) << endl;
    cout << setprecision(2) << setw(12) << Angle(x,DMM) << endl;
    cout << setprecision(2) << setw(12) << Angle(x,DMMm) << endl;
    cout << setprecision(2) << setw(12) << Angle(x,DMMSS) << endl;
    cout << setprecision(2) << setw(12) << Angle(x,DMMSSs) << endl;
}
```

Die Ausgabe stellt den Winkel $12^\circ 3456$ in fünf verschiedenen Formaten dar:

```

12.35
12 21
12 20.74
12 20 44
12 20 44.16
```

Ein eventuelles Vorzeichen würde linksbündig ausgegeben.

Als drittes Modul soll hier noch die Bibliothek `APC_VecMat3D` eingeführt werden, die zwei Klassen für dreidimensionale Vektoren und Matrizen sowie eine Vielzahl zugehöriger Operatoren und Funktionen bereitstellt. Dreidimensionale Vektoren dienen dazu, die Position eines Punktes \mathbf{r} im Raum zu beschreiben, wobei neben den kartesischen Koordinaten $\mathbf{r} = (x, y, z)$ auch die Polarkoordinaten r, ϑ und φ gebräuchlich sind (Abb. 2.1). Zwischen beiden Formen besteht der Zusammenhang

$$\begin{aligned}
x &= r \cos \vartheta \cos \varphi & r &= \sqrt{x^2 + y^2 + z^2} \\
y &= r \cos \vartheta \sin \varphi & \tan \varphi &= y/x \\
z &= r \sin \vartheta & \tan \vartheta &= z / \sqrt{x^2 + y^2}
\end{aligned} \tag{2.1}$$

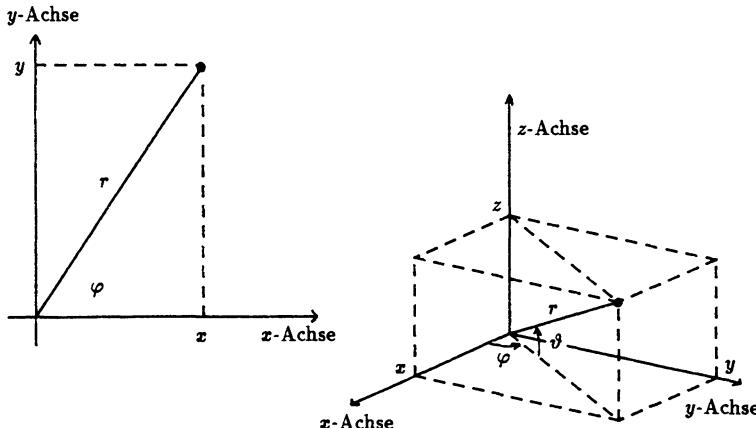


Abb. 2.1. Ebene und räumliche Polarkoordinaten

Die Klasse Vec3D erlaubt es, beide Darstellungen auf einfache Weise nebeneinander zu verwenden. Hierzu werden zunächst entsprechende Konstruktoren Vec3D() und Zugriffsoperatoren [] bereitgestellt:

```

//  

// Vec3D: dreidimensionale Vektoren  

//  

class Vec3D  

{  

public:  

    // Konstruktoren  

    Vec3D(); // Standardkonstruktor (initialisiert zum Nullvektor)  

    Vec3D(double X, double Y, double Z);  

    Vec3D(const Polar& polar);  

    // Lesender Zugriff auf Vektorkomponenten  

    double operator [] (index Index) const;  

    // Liefert die Polarwinkel oder den Betrag eines Vektors  

    double operator [] (pol_index Index);  

    // Einfache Ausgabe von Vektoren  

    friend ostream& operator << (ostream& os, const Vec3D& Vec);  

    // Weitere Operatoren und Funktionen  

    ...  

private:  

    // Datenelemente  

    double m_Vec[3];           // Komponenten des Vektors  

    double m_phi;               // Polarwinkel (Azimut)  

    double m_theta;              // Polarwinkel (Elevation)  

    double m_r;                 // Betrag des Vektors  

    bool m_bPolarValid;         // zeigt, ob Polarkomponenten gueltig sind  

    // Berechnung der polaren Komponenten bei Bedarf  

    void CalcPolarAngles();  

};
```

Die Datentypen `index`, `pol_index` und `Polar` sind dabei als

```
enum index { x=0, y=1, z=2 };
enum pol_index { phi=0, theta=1, r=2 }; // Azimut, Elevation, Radius
```

und

```
struct Polar {
    // Konstruktoren
    Polar();
    Polar(double Az, double Elev, double R = 1.0);
    // Datenelemente
    double phi; // Azimut des Vektors
    double theta; // Elevation des Vektors
    double r; // Betrag des Vektors
};
```

definiert. Die Umrechnung von kartesischen Koordinaten wird mit Hilfe der privaten Funktion

```
//  
// Berechne polare Komponenten  
//  
void Vec3D::CalcPolarAngles ()  
{  
    // Laenge der Projektion des Vektors in die x-y-Ebene  
    const double rhoSqr = m_Vec[0] * m_Vec[0] + m_Vec[1] * m_Vec[1];  
    // Betrag des Vektors  
    m_r = sqrt ( rhoSqr + m_Vec[2] * m_Vec[2] );  
    // Azimut des Vektors  
    if ( (m_Vec[0]==0.0) && (m_Vec[1]==0.0) )  
        m_phi = 0.0;  
    else  
        m_phi = atan2 (m_Vec[1], m_Vec[0]);  
    if ( m_phi < 0.0 ) m_phi += 2.0*pi;  
    // Elevation des Vektors  
    const double rho = sqrt ( rhoSqr );  
    if ( (m_Vec[2]==0.0) && (rho==0.0) )  
        m_theta = 0.0;  
    else  
        m_theta = atan2(m_Vec[2], rho);  
}
```

durchgeführt, sobald mit dem Operator `[]` erstmalig auf die Polarkoordinaten zugegriffen wird. Anschließend wird die Statusvariable `m_bPolarValid` auf `true` gesetzt, so daß eine Umrechnung jeweils nur einmal durchgeführt werden muß. Zu beachten ist, daß Winkel jeweils im Bogenmaß erwartet bzw. berechnet werden und bei der Ein- und Ausgabe gegebenenfalls umzurechnen sind. `APC_Math` stellt hierzu die Konstanten `Deg = 180°/π` und `Rad = π/180°` zur Verfügung.

Neben den elementaren Konstruktoren und Zugriffsoperatoren umfaßt die Klasse `Vec3D` eine Vielzahl von Operatoren und Funktionen, die es erlauben mit Vektoren auf einfache Weise zu rechnen. Vektoren können beispielsweise mit Hilfe des `+` Operators addiert werden, während der `*` Operator die Multiplikation eines Vektors mit einem Skalar erlaubt (vgl. Tabelle 2.1).

Tabelle 2.1. Vektor- und Matrixoperationen in APC_VecMat3D

Funktion	Arg ₁ (a)	Arg ₂ (b)	Wert (c)	Notation	Bedeutung
-	Vec3D		Vec3D	$c = -a$	Unäres Minus
	Mat3D		Mat3D	$C = -A$	
+	Vec3D	Vec3D	Vec3D	$c = a + b$	Vektor-Addition
	Mat3D	Mat3D	Mat3D	$C = A+B$	Matrix-Addition
*	double	Vec3D	Vec3D	$c = ab$	Skalar-Multiplikation
	Vec3D	double	Vec3D	$c = ab$	
	double	Mat3D	Mat3D	$C = aB$	
	Mat3D	double	Mat3D	$C = Ab$	
/	Vec3D	double	Vec3D	$c = a/b$	Skalar-Division
	Mat3D	double	Mat3D	$C = A/b$	
*	Mat3D	Vec3D	Vec3D	$c = Ab$	Matrix/Vektor Produkt
	Vec3D	Mat3D	Vec3D	$c = aB$	
Norm	Vec3D		double	$c = a $	Betrag
Dot	Vec3D	Vec3D	double	$c = a^T b$	Skalarprodukt
Cross	Vec3D	Vec3D	Vec3D	$c = a \times b$	Kreuzprodukt
Id3D			Mat3D	1	Einheitsmatrix
R_x	double		Mat3D	$R_x(a)$	Elementare Drehmatrix
R_y	double		Mat3D	$R_y(a)$	
R_z	double		Mat3D	$R_z(a)$	

Die Klasse Vec3D wird durch die Klasse Mat3D der dreidimensionalen Matrizen ergänzt, die wie folgt definiert ist:

```

// 
// Mat3D: dreidimensionale Transformationsmatrizen
//
class Mat3D
{
    public:
        // Konstruktoren (leere Matrix; Matrix aus Spaltenvektoren)
        Mat3D ();
        Mat3D ( const Vec3D& e_1, const Vec3D& e_2, const Vec3D& e_3 );
        // Zugriff auf Komponenten
        friend Vec3D Col(const Mat3D& Mat, index Index);
        friend Vec3D Row(const Mat3D& Mat, index Index);
        // Einfache Ausgabe von Matrizen
        friend ostream& operator << (ostream& os, const Mat3D& Mat);
        // Weitere Operatoren und Funktionen
        ...
    private:
        double m_Mat[3][3];    // Elemente der Matrix
};
```

Während Objekte der Vektorklasse – wie erwähnt – räumliche Koordinaten beschreiben, dienen Objekte der Matrixklasse im folgenden überwiegend zur Beschreibung räumlicher Drehungen von Koordinatensystemen. Einfache Drehungen

um die x -, y - oder z -Achse können mit Hilfe der Funktionen R_x , R_y und R_z erzeugt werden, die zu gegebenem Drehwinkel ϕ die Matrizen

$$\mathbf{R}_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & +\cos \phi & +\sin \phi \\ 0 & -\sin \phi & +\cos \phi \end{pmatrix} \quad (2.2)$$

$$\mathbf{R}_y(\phi) = \begin{pmatrix} +\cos \phi & 0 & -\sin \phi \\ 0 & 1 & 0 \\ +\sin \phi & 0 & +\cos \phi \end{pmatrix} \quad (2.3)$$

$$\mathbf{R}_z(\phi) = \begin{pmatrix} +\cos \phi & +\sin \phi & 0 \\ -\sin \phi & +\cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2.4)$$

berechnen. Die Vorzeichen sind hierbei so gewählt, daß ein positiver Winkel einer positiven Drehung (d.h. einer Drehung gegen den Uhrzeigersinn) des Bezugssystems um die Drehachse entspricht.

2.2 Kalender und julianisches Datum

In der Ephemeridenrechnung benötigt man häufig die Zeitdifferenz zwischen zwei gegebenen Daten. Dafür erweist sich eine kontinuierliche Tageszählung als sehr praktisch, wie sie in der Astronomie schon seit langem verwendet wird. Das julianische Datum gibt die Anzahl der Tage an, die seit dem 1. Januar des Jahres 4713 v.Chr. vergangen sind. Es ist nach Julius Scaliger benannt, dem Vater von Joseph Justus Scaliger, der es erstmals für chronologische Zwecke verwendete. Da die Zählung in biblischen Zeiträumen beginnt, nimmt das julianische Datum heute bereits recht hohe Werte an. Am Mittag des 23.7.1980 betrug es zum Beispiel 2444444.0 Tage. Bedenkt man, daß eine Sekunde nur etwa 0.00001 Tage lang ist, dann braucht man zur Angabe eines genauen Zeitpunkts schon ein zwölfstelliges julianisches Datum. Da sich die beiden ersten Ziffern aber über fast drei Jahrhunderte hinweg nicht verändern, verwendet man neben dem eigentlichen julianischen Datum auch das sogenannte *modifizierte julianische Datum*:

$$\text{MJD} = \text{JD} - 2400000.5$$

Es gibt die Zahl der seit dem 17.11.1858 0 Uhr vergangenen Tage an. Damit wechselt es wie das gewöhnliche Datum um Mitternacht und nicht am Mittag eines Tages wie das julianische Datum.

```
//-----
// Mjd: Bestimmt das Modifizierte Julianische Datum aus dem Kalenderdatum
//   Year,Month,Day   Komponenten des Kalenderdatums
//   Hour,Min,Sec     Komponenten der Zeit (optional)
//   <return>          Modifiziertes Julianisches Datum
//-----
double Mjd ( int Year,    int Month, int Day,
              int Hour=0, int Min=0, double Sec=0.0 );
```

```

double Mjd ( int Year, int Month, int Day,
              int Hour, int Min, double Sec )
{
    long      MjdMidnight;
    double   FracOfDay;
    int       b;
    if (Month<=2) { Month+=12; --Year;}
    if ( (10000L*Year+100L*Month+Day) <= 15821004L )
        b = -2 + ((Year+4716)/4) - 1179;      // Julianischer Kalender
    else
        b = (Year/400)-(Year/100)+(Year/4); // Gregorianischer Kalender
    MjdMidnight = 365L*Year - 679004L + b + int(30.6001*(Month+1)) + Day;
    FracOfDay   = Ddd(Hour,Min,Sec) / 24.0;
    return MjdMidnight + FracOfDay;
}

```

Die Funktion `Mjd` berücksichtigt, daß aufgrund der gregorianischen Kalenderreform auf den 4. Oktober 1582 (JD 2299159.5) der 15. Oktober (JD 2299160.5) folgte. Bis zu diesem Zeitpunkt galt der julianische Kalender, bei dem in jedem vierten Jahr ein 29. Februar eingeschoben wurde. Ab dem Reformdatum entfällt dieser Schalttag in jedem vollen Jahrhundert, dessen Jahreszahl nicht durch 400 teilbar ist. Die mittlere Jahreslänge des gregorianischen Kalenders beträgt demnach

$$365 + 1/4 - 1/100 + 1/400 = 365.2425 \text{ Tage}$$

und ist damit etwas kürzer als ein julianisches Jahr mit 365.25 Tagen.

Die Funktion `CalDat` berechnet aus dem modifizierten julianischen Datum das übliche Kalenderdatum. Sie wird wegen des engen Zusammenhangs mit dem Stoff dieses Abschnitts schon hier vorgestellt.

```

//-----
// CalDat: Kalenderdatum und Zeit aus dem Modifizierten Julianischen Datum
//   Mjd           Modifiziertes Julianisches Datum
//   Year,Month,Day Komponenten des Kalenderdatums
//   Hour          Zeit in Stunden in dezimaler Form
//-----
void CalDat ( double Mjd,
               int& Year, int& Month, int& Day, double & Hour )
{
    long      a,b,c,d,e,f;
    double   FracOfDay;
    // Rechne julianische Tageszahl in Kalenderdatum um
    a = long(Mjd+2400001.0);
    if ( a < 2299161 ) { // Julianischer Kalender
        b = 0;
        c = a + 1524;
    }
    else { // Gregorianischer Kalender
        b = long((a-1867216.25)/36524.25);
        c = a + b - (b/4) + 1525;
    }
    d      = long ( (c-122.1)/365.25 );

```

```

e      = 365*d + d/4;
f      = long ( (c-e)/30.6001 );
Day   = c - e - int(30.6001*f);
Month = f - 1 - 12*(f/14);
Year  = d - 4715 - ((7+Month)/10);
FracOfDay = Mjd - floor(Mjd);
Hour  = 24.0*FracOfDay;
}
//-----
// CalDat: Kalenderdatum und Zeit aus dem Modifizierten Julianischen Datum
// Mjd           Modifiziertes Julianisches Datum
// Year,Month,Day Komponenten des Kalenderdatums
// Hour,Min,Sec  Komponenten der Zeit
//-----
void CalDat ( double Mjd,
              int& Year, int& Month, int& Day,
              int& Hour, int& Min, double& Sec )
{
    double Hours;
    CalDat (Mjd, Year, Month, Day, Hours);
    DMS (Hours, Hour, Min, Sec);
}

```

Die beiden überladenen Versionen von CalDat unterscheiden sich in der Behandlung des Tagesbruchteils seit Mitternacht, der entweder in dezimalen Stunden oder in Stunden, Minuten und Sekunden zur Verfügung gestellt wird. Ähnlich wie bei der sexagesimalen Darstellung von Winkeln ist die zuletzt genannte Form meist nur bei Ausgaben von Zeitpunkten erforderlich. Das APC_Time Modul stellt deshalb neben Mjd und CalDat zwei spezielle Klassen Time und DateTime mit entsprechenden Konstruktoren

```

Time ( double Hour, TimeFormat Format=HHMMSS );
DateTime ( double Mjd, TimeFormat Format=None );

```

zur Verfügung, die diese Aufgabe noch besser unterstützen. Der Aufzählungstyp

```

enum TimeFormat {
    None,    // keine Ausgabe der Zeit (nur Datum)
    DDd,    // Ausgabe der Zeit als Tagesbruchteil
    HHh,    // Ausgabe der Zeit in Stunden mit einer Nachkommastelle
    HHMM,   // Zeit in Stunden und Minuten (Rundung auf naechste Minute)
    HHMMSS // Zeit in Stunden, Minuten und Sekunden (Rundung naechste Sekunde)
};

```

dient dabei zur Festlegung des Formats und der notwendigen Rundungen bei der Ausgabe von Time und DateTime Objekten mittels des Schiebeoperators (<<). Hierzu ein Beispiel:

```

double MJD = Mjd ( 1961,01,14, 03,30,10.0 ); // 14. Jan. 1961, 3:30:10 Uhr
double JD = MJD + 2400000.5;                  // Julianisches Datum
cout << setprecision(1)
     << "Date:" << DateTime(MJD,HHMMSS)        // Ausgabe
     << setprecision(3)
     << "MJD: " << setw(12) << MJD
     << "JD: " << setw(12) << JD;

```

2.3 Ekliptik und Äquator

Ekliptikale und äquatoriale Koordinaten unterscheiden sich hinsichtlich der Ebene, bezüglich derer sie gemessen werden. Im einen Fall handelt es sich bei der x - y -Ebene um die Erdbahnebene (*Ekliptik*), im anderen Fall um die Ebene senkrecht zur Erdachse, also parallel zur Äquatorebene der Erde (Abb. 2.2). Die gemeinsame x - x' -Richtung wird als Richtung zum Frühlingspunkt oder kurz nur als *Frühlingspunkt* (Υ) bezeichnet. Sie steht senkrecht auf den Richtungen zum Nordpol der Ekliptik (z -Achse) und zum Himmelsnordpol (z' -Achse). Der Winkel ε zwischen Ekliptik und Äquator beträgt rund $23^\circ 5'$.

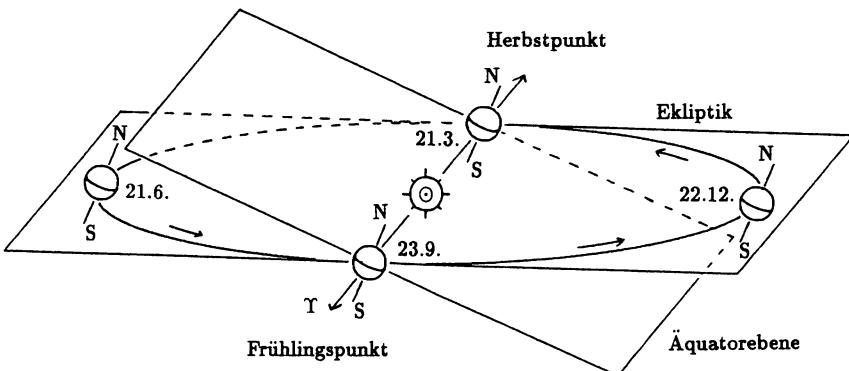


Abb. 2.2. Ekliptik und Äquator

Wie lautet nun der Zusammenhang zwischen den ekliptikalnen Koordinaten $r = (x, y, z)$ und den äquatorialen Koordinaten $r' = (x', y', z')$ eines Punktes? Wegen $x = x'$ genügt es, die Transformation $(y, z) \leftrightarrow (y', z')$ zu untersuchen. Zunächst hat ein Punkt auf der y -Achse ($z = 0$) die äquatorialen Koordinaten $y' = +y \cos \varepsilon$ und $z' = +y \sin \varepsilon$. Liegt der Punkt auf der z -Achse ($y = 0$), dann lauten seine äquatorialen Koordinaten $y' = -z \sin(\varepsilon)$ und $z' = +z \cos \varepsilon$. Zusammengefäßt hat ein beliebiger Punkt (x, y, z) im Äquatorsystem die Komponenten

$$\begin{aligned} x' &= +x \\ y' &= +y \cos \varepsilon - z \sin \varepsilon \\ z' &= +y \sin \varepsilon + z \cos \varepsilon . \end{aligned} \quad (2.5)$$

Ganz entsprechend erhält man die Umkehrung

$$\begin{aligned} x &= +x' \\ y &= +y' \cos \varepsilon + z' \sin \varepsilon \\ z &= -y' \sin \varepsilon + z' \cos \varepsilon \end{aligned} \quad (2.6)$$

Die zu (x, y, z) gehörigen Polarkoordinaten bezeichnet man als ekliptikale Länge l , ekliptikale Breite b und Entfernung r (vgl. Abb. 2.3).

$$\begin{aligned} x &= r \cos b \cos l & x' &= r \cos \delta \cos \alpha \\ y &= r \cos b \sin l & y' &= r \cos \delta \sin \alpha \\ z &= r \sin b & z' &= r \sin \delta . \end{aligned} \quad (2.7)$$

Im äquatorialen System entsprechen diesen Koordinaten die Rektaszension α , die Deklination δ und wieder die Entfernung r , die in beiden Systemen den gleichen Wert hat.

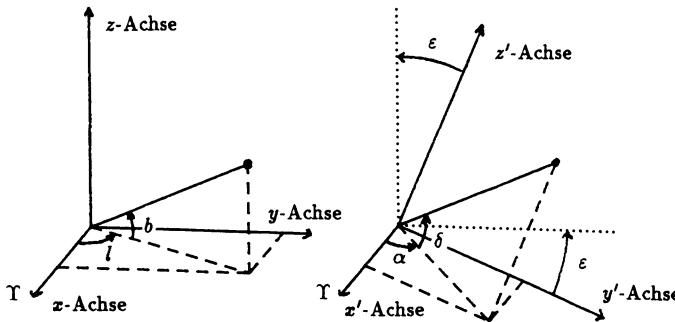


Abb. 2.3. Ekliptikale und äquatoriale Koordinaten

Besonders kompakt und übersichtlich lässt sich die Transformation zwischen ekliptikal en und äquatorialen Koordinaten in Vektor-/Matrix-Schreibweise darstellen. Unter Verwendung der weiter oben eingeführten elementaren Drehmatrizen gilt:

$$\mathbf{r} = \mathbf{R}_x(+\varepsilon) \mathbf{r}' \quad \text{und} \quad \mathbf{r}' = \mathbf{R}_x^T(+\varepsilon) \mathbf{r} = \mathbf{R}_x(-\varepsilon) \mathbf{r} , \quad (2.8)$$

wobei \mathbf{R}_x^T die Transponierte (Spiegelung) der Matrix \mathbf{R}_x bezeichnet. Analog hierzu erhält man die transformierten Koordinaten eines Vektors vom Typ Vec3D durch Multiplikation mit der entsprechenden Drehmatrix. Hierzu wieder ein kurzes Programmbeispiel:

```
double eps = Rad*23.5; // Ekliptikschiefe in [rad]
Vec3D e = Vec3D(1.0,2.0,3.0); // Ekliptikale Koordinaten
Vec3D a = R_x(eps)*e; // Aequatoriale Koordinaten
cout << "1 " << Deg*e[phi] << " b " << Deg*e[theta] << endl
<< "RA " << Deg*a[phi]/15.0 << " Dec " << Deg*a[theta] << endl;
```

Der Wert der Rektaszension wird dabei – wie allgemein üblich – in Stunden ($1^h \equiv 15^\circ$) statt in Grad angegeben.

Bisher wurde noch nicht über den genauen Wert der Ekliptikschiefe ε gesprochen. ε ist nicht konstant, sondern verringert sich um rund $47''$ pro Jahrhundert. Dies liegt im wesentlichen daran, daß die Erdbahn sich durch die Anziehungskräfte der übrigen Planeten langsam verlagert. Will man bei der Koordinatentransformation keine Fehler machen, dann spielt es also durchaus eine Rolle, ob man sich auf die Ekliptik des Jahres 1950 oder des Jahres 2000 bezieht. Man bezeichnet diese Jahresangabe als Äquinoktium der Koordinaten. Der genaue Wert der Ekliptikschiefe als Funktion der Zeit beträgt

$$\varepsilon = 23^\circ 43' 929111 - 46.''8150 T - 0.''00059 T^2 + 0.''001813 T^3 , \quad (2.9)$$

wobei T die Zahl der julianischen Jahrhunderte zwischen dem Äquinoktium und dem 1.1.2000 (12 Uhr) bezeichnet. Im Gegensatz zum Jahrhundert des gregorianischen Kalenders mit im Mittel 36524.25 Tagen ist ein julianisches Jahrhundert immer genau 36525 Tage lang. Wichtige julianische Epochen sind J1900 (0.5 Jan. 1900, JD 2415020.0) und J2000 (1.5 Jan. 2000, JD 2451545.0). Zwischen beiden liegt gerade ein volles julianisches Jahrhundert. T berechnet sich für ein Äquinoktium mit dem julianischen Datum JD damit nach der Formel

$$T = \frac{JD - 2451545}{36525} .$$

Näherungsweise erhält man T für das Jahr y zu $T = (y - 2000)/100$. Die folgende Funktion liefert zu einer gegebenen Epoche T direkt die Matrix für die Transformation von äquatorialen in ekliptikale Koordinaten.

```
//-----
// Equ2EclMatrix: Transformation aequatorialer in ekliptikale Koordinaten
//   T          Zeit in Julianischen Jahrhunderten seit J2000
//   <return>  Transformationsmatrix
//-----
Mat3D Equ2EclMatrix (double T)
{
    const double
        eps = ( 23.43929111-(46.8150+(0.00059-0.001813*T)*T)*T/3600.0 ) * Rad;
    return R_x(eps);
}
```

Die umgekehrte Transformation wird entsprechend durch die transponierte Matrix

```
//-----
// Ecl2EquMatrix: Transformation ekliptikaler in aequatoriale Koordinaten
//   T          Zeit in Julianischen Jahrhunderten seit J2000
//   <return>  Transformationsmatrix
//-----
Mat3D Ecl2EquMatrix (double T)
{
    return Transp(Equ2EclMatrix(T));
}
```

beschrieben. Mit Hilfe von `Ecl2EquMatrix` kann man das obige Beispiel folgendermaßen schreiben (Äquinoktium sei jetzt J1950 = JD 2433282.50):

```
double MJD = 33282.0;                      // Epoche
double T   = (MJD-MJD_J2000)/36525.0;
Mat3D U   = Ecl2EquMatrix(T);                // Transformationsmatrix
Vec3D e   = Vec3D(1.0,2.0,3.0);             // Ekliptikale Koordinaten
Vec3D a   = U*e;                            // Aequatoriale Koordinaten
cout << "l " << Deg*e[phi]    << " b " << Deg*e[theta] << endl
     << "RA " << Deg*a[phi]/15.0 << " Dec " << Deg*a[theta] << endl;
```

2.4 Präzession

Die Verschiebung der Erdachse und der Ekliptik durch die Kräfte von Sonne, Mond und Planeten führt nicht nur zu der leichten Änderung des Winkels ε zwischen Äquator und Ekliptik, sondern vor allem zu einer Verschiebung des Frühlingspunktes um etwa $1^\circ 5$ pro Jahrhundert ($1'$ pro Jahr). Das Äquinoktium der verwendeten Koordinatensysteme muß deshalb für eine genaue Rechnung mit angegeben werden. In Gebrauch sind vor allem

- das Äquinoktium des Datums,
- das Äquinoktium J2000 und
- das Äquinoktium B1950.

Äquinoktium des Datums bedeutet, daß man sich auf Äquator, Ekliptik und Frühlingspunkt des gerade aktuellen Datums bezieht. Ein solcher täglicher Wechsel des Koordinatensystems ist zum Beispiel dann sinnvoll, wenn man die Koordinaten eines Planeten für die Arbeit an den Teilkreisen eines parallaktisch montierten Fernrohrs oder an einem Meridiankreis benötigt. Durch die Verlagerung der Erdachse wird ja auch die Orientierung der Stundenachse des Fernrohrs verändert. Will man dagegen die tatsächliche räumliche Bewegung des Planeten studieren, dann arbeitet man besser mit einem festen Äquinoktium, etwa dem der julianischen Epoche J2000 (1.5 Januar 2000 = JD 2451545.0), das seit 1984 allgemein eingeführt ist. Das ältere Äquinoktium B1950 war davor lange in Gebrauch und wurde für viele Sternkataloge und Atlanten verwendet (z.B. SAO Star Catalog und Atlas Coeli). Der Vorsatz B deutet an, daß es sich nicht um die Mitte zwischen den Epochen J1900 und J2000 handelt (dies wäre nämlich J1950 = JD 2433282.5), sondern um den Beginn des Besseljahres 1950 (0.923 Jan. 1950 = JD 2433282.423).

Die Transformation der Koordinaten von einem Äquinoktium zum anderen erfordert ähnliche Schritte wie die von ekliptikalnen Koordinaten in äquatoriale. Während dort die beiden Koordinatensysteme durch eine einzige Drehung um die x -Achse auseinander hervorgingen, hat man es allerdings nun mit insgesamt drei Drehungen um die z -Achse, dann die x -Achse und schließlich noch einmal die z -Achse zu tun.

Betrachtet man zunächst die Ekliptik zur Zeit T_0 und zur Zeit $T_0 + T$, dann schließen diese beiden Ebenen miteinander den Winkel

$$\begin{aligned} \pi &= (47.^{\circ}0029 - 0.^{\circ}06603 \cdot T_0 + 0.^{\circ}000598 \cdot T_0^2) \cdot T \\ &\quad + (-0.^{\circ}03302 + 0.^{\circ}000598 \cdot T_0) \cdot T^2 + 0.^{\circ}000060 \cdot T^3 \end{aligned} \tag{2.10}$$

ein. Legt man zwei Koordinatensysteme (x', y', z') und (x'', y'', z'') so, daß die x' - y' -Ebene mit der Ekliptik zur Zeit T_0 und die x'' - y'' -Ebene mit der Ekliptik zur Zeit $T_0 + T$ zusammenfällt, dann gilt:

$$\begin{aligned} x'' &= x' \\ y'' &= +\cos \pi \cdot y' + \sin \pi \cdot z' \\ z'' &= -\sin \pi \cdot y' + \cos \pi \cdot z' \quad . \end{aligned}$$

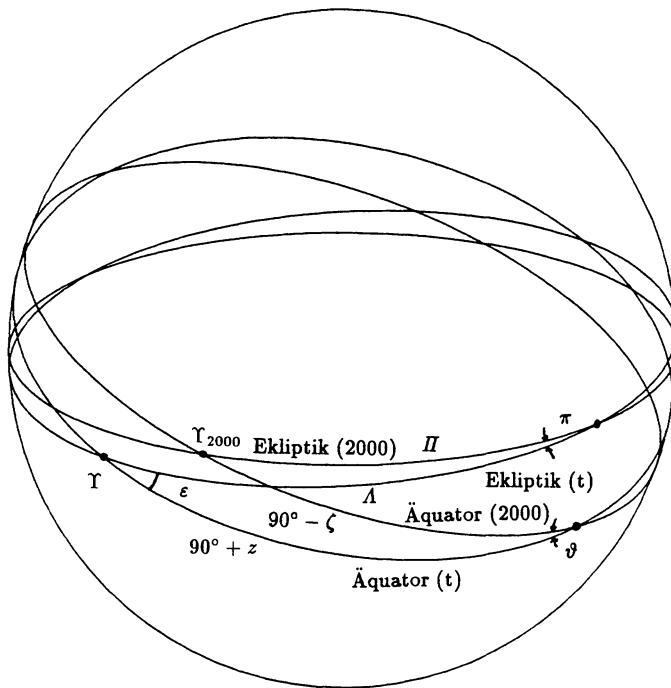


Abb. 2.4. Der Einfluß der Präzession auf Ekliptik, Äquator und Frühlingspunkt

Hierbei wurde vorausgesetzt, daß die x' - und die x'' -Achse in Richtung der gemeinsamen Schnittlinie beider Ebenen orientiert sind. Es seien nun (x_0, y_0, z_0) die ekliptikalnen Koordinaten zum Äquinoktium T_0 und (x, y, z) die Koordinaten zum Äquinoktium $T_0 + T$. Dann gilt

$$\begin{aligned} x' &= +\cos \Pi \cdot x_0 + \sin \Pi \cdot y_0 \\ y' &= -\sin \Pi \cdot x_0 + \cos \Pi \cdot y_0 \\ z' &= z_0 \end{aligned}$$

und

$$\begin{aligned} x &= +\cos \Lambda \cdot x'' - \sin \Lambda \cdot y'' \\ y &= +\sin \Lambda \cdot x'' + \cos \Lambda \cdot y'' \\ z &= z'' , \end{aligned}$$

mit

$$\begin{aligned} \Pi &= (174^\circ 876383889 + 3289.''4789T_0 + 0.''60622T_0^2) \\ &\quad + (-869.''8089 - 0.''50491T_0)T + 0.''03536T^2 \\ p &= (5029.''0966 + 2.''22226T_0 - 0.''000042T_0^2)T \\ &\quad + (1.''11113 - 0.''000042T_0)T^2 - 0.''000006T^3 \\ \Lambda &= \Pi + p . \end{aligned} \tag{2.11}$$

Π und Λ sind die Winkel zwischen der x'/x'' -Achse und dem Frühlingspunkt Υ_0 zum Äquinoktium T_0 (x_0 -Achse) bzw. dem Frühlingspunkt Υ zur Zeit $T_0 + T$ (x -Achse). p wird als Präzession in Länge bezeichnet, weil es im wesentlichen die Verschiebung des Frühlingspunktes in ekliptikal Länge darstellt.

Setzt man diese drei Beziehungen ineinander ein, dann erhält man zusammengefaßt die Gleichung

$$\mathbf{r} = \mathbf{P} \mathbf{r}_0 \quad \text{mit} \quad \mathbf{P} = \mathbf{R}_z(-\Pi - p) \mathbf{R}_x(\pi) \mathbf{R}_z(\Pi) \quad . \quad (2.12)$$

Ausgeschrieben lauten die Elemente von \mathbf{P}

$$p_{11} = + \cos \Lambda \cos \Pi + \sin \Lambda \cos \pi \sin \Pi$$

$$p_{21} = + \sin \Lambda \cos \Pi - \cos \Lambda \cos \pi \sin \Pi$$

$$p_{31} = + \sin \pi \sin \Pi$$

$$p_{12} = + \cos \Lambda \sin \Pi - \sin \Lambda \cos \pi \cos \Pi$$

$$p_{22} = + \sin \Lambda \sin \Pi + \cos \Lambda \cos \pi \cos \Pi \quad (2.13)$$

$$p_{32} = - \sin \pi \cos \Pi$$

$$p_{13} = - \sin \Lambda \sin \pi$$

$$p_{23} = + \cos \Lambda \sin \pi$$

$$p_{33} = + \cos \pi \quad .$$

In äquatorialen Koordinaten entsprechen den drei Winkeln π , Π und Λ die Winkel $90^\circ - \zeta$, ϑ und $90^\circ + z$:

$$\begin{aligned} \zeta &= (2306.^{\circ}2181 + 1.^{\circ}39656T_0 - 0.^{\circ}000139T_0^2)T \\ &\quad + (0.^{\circ}30188 - 0.^{\circ}000345T_0)T^2 + 0.^{\circ}017998T^3 \end{aligned}$$

$$\begin{aligned} \vartheta &= (2004.^{\circ}3109 - 0.^{\circ}85330T_0 - 0.^{\circ}000217T_0^2)T \\ &\quad + (-0.^{\circ}42665 - 0.^{\circ}000217T_0)T^2 - 0.^{\circ}041833T^3 \end{aligned} \quad (2.14)$$

$$z = \zeta + (0.^{\circ}79280 + 0.^{\circ}000411T_0)T^2 + 0.^{\circ}000205T^3 \quad .$$

Dies führt auf entsprechende Formeln mit

$$\mathbf{P} = \mathbf{R}_z(-z) \mathbf{R}_y(\vartheta) \mathbf{R}_z(-\zeta) \quad (2.15)$$

oder, komponentenweise,

$$p_{11} = - \sin z \sin \zeta + \cos z \cos \vartheta \cos \zeta$$

$$p_{21} = + \cos z \sin \zeta + \sin z \cos \vartheta \cos \zeta$$

$$p_{31} = + \sin \vartheta \cos \zeta$$

$$p_{12} = - \sin z \cos \zeta - \cos z \cos \vartheta \sin \zeta$$

$$p_{22} = + \cos z \cos \zeta - \sin z \cos \vartheta \sin \zeta \quad (2.16)$$

$$p_{32} = - \sin \vartheta \sin \zeta$$

$$p_{13} = - \cos z \sin \vartheta$$

$$p_{23} = - \sin z \sin \vartheta$$

$$p_{33} = + \cos \vartheta \quad .$$

Den aufwendigsten Teil bei der Berechnung der Präzession bildet die Bestimmung der verschiedenen Winkelgrößen und der Matrixelemente p_{ij} . Die Transformationsmatrizen hängen aber nur von den beiden Äquinoktien T_0 und $T_0 + T$ ab und können daher wiederverwendet werden, wenn man eine ganze Reihe verschiedener Positionen von einem festen Äquinoktium zum anderen transformieren möchte.

```
-----
// PrecMatrix_Ecl: Praezession ekliptikaler Koordinaten
//   T1      Gegebene Epoche
//   T2      Epoche, zu der hin umgerechnet werden soll
//   <return> Transformationsmatrix fuer Praezession
// Beachte: T1 and T2 in Julianischen Jahrhunderten seit J2000
-----
Mat3D PrecMatrix_Ecl (double T1, double T2)
{
    const double dT = T2-T1;
    double Pi, pi, p_a;

    Pi = 174.876383889*Rad +
        ((3289.4789+0.60622*T1)*T1) +
        ((-869.8089-0.50491*T1) + 0.03536*dT)*dT )/Arcs;
    pi = ( (47.0029-(0.06603-0.000598*T1)*T1) +
        ((-0.03302+0.000598*T1)+0.000060*dT)*dT )*dT/Arcs;
    p_a = ( (5029.0966+(2.22226-0.000042*T1)*T1) +
        ((1.11113-0.000042*T1)-0.000006*dT)*dT )*dT/Arcs;
    return R_z(-(Pi+p_a)) * R_x(pi) * R_z(Pi);
}

-----
// PrecMatrix_Equ: Praezession aequatorialer Koordinaten
//   T1      Gegebene Epoche
//   T2      Epoche, zu der hin umgerechnet werden soll
//   <return> Transformationsmatrix fuer Praezession
// Beachte: T1 and T2 in Julianischen Jahrhunderten seit J2000
-----
Mat3D PrecMatrix_Equ (double T1, double T2)
{
    const double dT = T2-T1;
    double zeta,z,theta;

    zeta = ( (2306.2181+(1.39656-0.000139*T1)*T1) +
        ((0.30188-0.000344*T1)+0.017998*dT)*dT )*dT/Arcs;
    z = zeta + ( (0.79280+0.000411*T1)+0.000205*dT)*dT*dT/Arcs;
    theta = ( (2004.3109-(0.85330+0.000217*T1)*T1) -
        ((0.42665+0.000217*T1)+0.041833*dT)*dT )*dT/Arcs;
    return R_z(-z) * R_y(theta) * R_z(-zeta);
}
```

Um ein Gefühl für den Einfluß der Präzession zu gewinnen und die Verwendung der verschiedenen Funktionen einzubüben, folgt hier ein weiteres Beispiel. Aus einem Satz ekliptikaler Koordinaten zum Äquinoktium B1950 sollen die zugehörigen äquatorialen Koordinaten zum Äquinoktium J2000 berechnet werden.

Dies kann auf zwei Wegen geschehen, je nachdem, ob man die Präzession oder den Wechsel Ekliptik-Äquator zuerst behandelt.

```

// Deklarationen
double T_B1950 = -0.500002108;           // Epoche B1950 (JD 2433282.423)
double l    = 200.0*Rad;                  // Laenge
double b    = 10.0*Rad;                   // Breite
Vec3D r_0 = Vec3D(Polar(l,b));          // Ekliptikale Koordinaten B1950
Vec3D r;

// Weg 1
r = PrecMatrixEcl(T_B1950,T_J2000) * r_0; // Ekliptik J2000
cout << "l,b (J2000) " << Deg*r[phi]      << Deg*r[theta] << endl;
r = Ecl2EquMatrix(T_J2000) * r;            // Aequator J2000
cout << "RA,Dec (J2000) " << Deg*r[phi]/15.0 << Deg*r[theta] << endl;

// Weg 2
r = Ecl2EquMatrix(T_B1950) * r_0;          // Aequator B1950
cout << "RA,Dec (B1950) " << Deg*r[phi]/15.0 << Deg*r[theta] << endl;
r = PrecMatrixEqu(T_B1950,T_J2000) * r;    // Aequator J2000
cout << "RA,Dec (J2000) " << Deg*r[phi]/15.0 << Deg*r[theta] << endl;

```

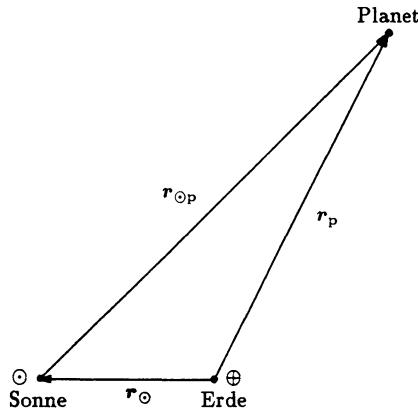


Abb. 2.5. Das Dreieck Erde-Sonne-Planet

2.5 Geozentrische Koordinaten und die Sonnenbahn

Zur Vervollständigung des Programms fehlt nun noch die Möglichkeit, heliozentrische (auf den Mittelpunkt der Sonne bezogene) und geozentrische (auf den Mittelpunkt der Erde bezogene) Koordinaten ineinander umzuwandeln. Dieser Wechsel des Koordinatenursprungs wird durch die Gleichungen

$$\mathbf{r}_p = \mathbf{r}_{\odot p} + \mathbf{r}_\odot \quad \text{und} \quad \mathbf{r}_{\odot p} = \mathbf{r}_p - \mathbf{r}_\odot \quad (2.17)$$

beschrieben, die sich aus dem in Abb. 2.5 dargestellten Dreieck ergeben. $\mathbf{r}_{\odot p}$ und \mathbf{r}_p sind darin der heliozentrische und der geozentrische Ortsvektor eines Punktes

P , während \mathbf{r}_\odot den geozentrischen Ort der Sonne bezeichnet. Komponentenweise geschrieben lauten die Transformationsformeln

$$\begin{aligned} x_p &= x_{\odot p} + x_\odot & x_{\odot P} &= x_P - x_\odot \\ y_p &= y_{\odot p} + y_\odot & \text{und} & y_{\odot P} = y_P - y_\odot \\ z_p &= z_{\odot p} + z_\odot & z_{\odot P} &= z_P - z_\odot \end{aligned},$$

wobei es keine Rolle spielt, ob man zur Darstellung der verschiedenen Vektoren ekliptikale oder äquatoriale Koordinaten verwendet.

Da sich jeder Fehler in der Position der Sonne unweigerlich auf die Genauigkeit der Umrechnung auswirkt, lohnt es sich, einige Mühe auf die Berechnung der Sonnenkoordinaten zu verwenden. Die Funktion `SunPos`, die hier eingesetzt wird, erreicht eine Genauigkeit von rund $1''$, was für die meisten Zwecke ausreichen sollte. Sie liefert zu einem Zeitpunkt

$$T = (\text{JD} - 2451545)/36525$$

die rechtwinkeligen ekliptikalnen Koordinaten

$$\mathbf{r}_\odot = \begin{pmatrix} x_\odot \\ y_\odot \\ z_\odot \end{pmatrix} = \begin{pmatrix} R \cos B \cos L \\ R \cos B \sin L \\ R \sin B \end{pmatrix} . \quad (2.18)$$

Diese beziehen sich wie die zugehörige ekliptikale Länge (L) und Breite (B) der Sonne auf das Äquinoktium des Datums, können aber mit den bereits behandelten Formeln jederzeit auf ein anderes Äquinoktium übertragen werden.

```
-----
// SunPos: Berechnet die ekliptikale Position der Sonne unter Verwendung einer
// analytischen Reihenentwicklung
// T          Zeit in Julianischen Jahrhunderten seit J2000
// <return>   Geozentrische Position der Sonne (in [AE]), bezogen auf
//             Ekliptik und Fruehlingspunkt des Datums
//-----
Vec3D SunPos (double T)
{
    // Variablen
    double M2,M3,M4,M5,M6;           // Mittlere Anomalien
    double D, A, U;                 // Mittlere Argumente der Mondbahn
    Pert Ven, Mar, Jup, Sat;        // Stoerungen
    double dl, dr, db;              // Korrekturen in Laenge ["],
                                    // Entfernung [AE] und Breite ["]
    double l,b,r;                  // Ekliptikale Koordinaten

    // Mittlere Anomalien der Planeten und mittlere Argumente der Mondbahn [rad]
    M2 = pi2 * Frac ( 0.1387306 + 162.5485917*T );
    M3 = pi2 * Frac ( 0.9931266 + 99.9973604*T );
    M4 = pi2 * Frac ( 0.0543250 + 53.1666028*T );
    M5 = pi2 * Frac ( 0.0551750 + 8.4293972*T );
    M6 = pi2 * Frac ( 0.8816500 + 3.3938722*T );
    D = pi2 * Frac ( 0.8274 + 1236.8531*T );
    A = pi2 * Frac ( 0.3749 + 1325.5524*T );
    U = pi2 * Frac ( 0.2591 + 1342.2278*T );
```

```

// Keplerbewegung und Stoerungen durch Venus
Ven.Init ( T, M3,0,7, M2,-6,0 );
Ven.Term ( 1, 0,0,-0.22,6892.76,-16707.37, -0.54, 0.00, 0.00 );
Ven.Term ( 1, 0,1,-0.06, -17.35, 42.04, -0.15, 0.00, 0.00 );
Ven.Term ( 1, 0,2,-0.01, -0.05, 0.13, -0.02, 0.00, 0.00 );
Ven.Term ( 2, 0,0, 0.00, 71.98, -139.57, 0.00, 0.00, 0.00 );
Ven.Term ( 2, 0,1, 0.00, -0.36, 0.70, 0.00, 0.00, 0.00 );
Ven.Term ( 3, 0,0, 0.00, 1.04, -1.75, 0.00, 0.00, 0.00 );
Ven.Term ( 0,-1,0, 0.03, -0.07, -0.16, -0.07, 0.02,-0.02 );
Ven.Term ( 1,-1,0, 2.35, -4.23, -4.75, -2.64, 0.00, 0.00 );
Ven.Term ( 1,-2,0,-0.10, 0.06, 0.12, 0.20, 0.02, 0.00 );
Ven.Term ( 2,-1,0,-0.06, -0.03, 0.20, -0.01, 0.01,-0.09 );
Ven.Term ( 2,-2,0,-4.70, 2.90, 8.28, 13.42, 0.01,-0.01 );
Ven.Term ( 3,-2,0, 1.80, -1.74, -1.44, -1.57, 0.04,-0.06 );
Ven.Term ( 3,-3,0,-0.67, 0.03, 0.11, 2.43, 0.01, 0.00 );
Ven.Term ( 4,-2,0, 0.03, -0.03, 0.10, 0.09, 0.01,-0.01 );
Ven.Term ( 4,-3,0, 1.51, -0.40, -0.88, -3.36, 0.18,-0.10 );
Ven.Term ( 4,-4,0,-0.19, -0.09, -0.38, 0.77, 0.00, 0.00 );
Ven.Term ( 5,-3,0, 0.76, -0.68, 0.30, 0.37, 0.01, 0.00 );
Ven.Term ( 5,-4,0,-0.14, -0.04, -0.11, 0.43,-0.03, 0.00 );
Ven.Term ( 5,-5,0,-0.05, -0.07, -0.31, 0.21, 0.00, 0.00 );
Ven.Term ( 6,-4,0, 0.15, -0.04, -0.06, -0.21, 0.01, 0.00 );
Ven.Term ( 6,-5,0,-0.03, -0.03, -0.09, 0.09,-0.01, 0.00 );
Ven.Term ( 6,-6,0, 0.00, -0.04, -0.18, 0.02, 0.00, 0.00 );
Ven.Term ( 7,-5,0,-0.12, -0.03, -0.08, 0.31,-0.02,-0.01 );
dl = Ven.dl(); dr = Ven.dr(); db = Ven.db();

// Stoerungen durch Mars
Mar.Init ( T, M3,1,5, M4,-8,-1 );
Mar.Term ( 1,-1,0,-0.22, 0.17, -0.21, -0.27, 0.00, 0.00 );
Mar.Term ( 1,-2,0,-1.66, 0.62, 0.16, 0.28, 0.00, 0.00 );
Mar.Term ( 2,-2,0, 1.96, 0.57, -1.32, 4.55, 0.00, 0.01 );
Mar.Term ( 2,-3,0, 0.40, 0.15, -0.17, 0.46, 0.00, 0.00 );
Mar.Term ( 2,-4,0, 0.53, 0.26, 0.09, -0.22, 0.00, 0.00 );
Mar.Term ( 3,-3,0, 0.05, 0.12, -0.35, 0.15, 0.00, 0.00 );
Mar.Term ( 3,-4,0,-0.13, -0.48, 1.06, -0.29, 0.01, 0.00 );
Mar.Term ( 3,-5,0,-0.04, -0.20, 0.20, -0.04, 0.00, 0.00 );
Mar.Term ( 4,-4,0, 0.00, -0.03, 0.10, 0.04, 0.00, 0.00 );
Mar.Term ( 4,-5,0, 0.05, -0.07, 0.20, 0.14, 0.00, 0.00 );
Mar.Term ( 4,-6,0,-0.10, 0.11, -0.23, -0.22, 0.00, 0.00 );
Mar.Term ( 5,-7,0,-0.05, 0.00, 0.01, -0.14, 0.00, 0.00 );
Mar.Term ( 5,-8,0, 0.05, 0.01, -0.02, 0.10, 0.00, 0.00 );
dl += Mar.dl(); dr += Mar.dr(); db += Mar.db();

// Stoerungen durch Jupiter
Jup.Init ( T, M3,-1,3, M5,-4,-1 );
Jup.Term (-1,-1,0, 0.01, 0.07, 0.18, -0.02, 0.00,-0.02 );
Jup.Term ( 0,-1,0,-0.31, 2.58, 0.52, 0.34, 0.02, 0.00 );
Jup.Term ( 1,-1,0,-7.21, -0.06, 0.13,-16.27, 0.00,-0.02 );
Jup.Term ( 1,-2,0,-0.54, -1.52, 3.09, -1.12, 0.01,-0.17 );
Jup.Term ( 1,-3,0,-0.03, -0.21, 0.38, -0.06, 0.00,-0.02 );
Jup.Term ( 2,-1,0,-0.16, 0.05, -0.18, -0.31, 0.01, 0.00 );
Jup.Term ( 2,-2,0, 0.14, -2.73, 9.23, 0.48, 0.00, 0.00 );
Jup.Term ( 2,-3,0, 0.07, -0.55, 1.83, 0.25, 0.01, 0.00 );

```

```

Jup.Term ( 2,-4,0, 0.02, -0.08,      0.25,  0.06, 0.00, 0.00);
Jup.Term ( 3,-2,0, 0.01, -0.07,      0.16,  0.04, 0.00, 0.00);
Jup.Term ( 3,-3,0,-0.16, -0.03,      0.08, -0.64, 0.00, 0.00);
Jup.Term ( 3,-4,0,-0.04, -0.01,      0.03, -0.17, 0.00, 0.00);
dl += Jup.dl(); dr += Jup.dr(); db += Jup.db();

// Stoerungen durch Saturn
Sat.Init ( T, M3,0,2, M6,-2,-1 );
Sat.Term ( 0,-1,0, 0.00, 0.32,      0.01,  0.00, 0.00, 0.00);
Sat.Term ( 1,-1,0,-0.08, -0.41,      0.97, -0.18, 0.00,-0.01);
Sat.Term ( 1,-2,0, 0.04, 0.10,      -0.23,  0.10, 0.00, 0.00);
Sat.Term ( 2,-2,0, 0.04, 0.10,      -0.35,  0.13, 0.00, 0.00);
dl += Sat.dl(); dr += Sat.dr(); db += Sat.db();

// Abstand des Schwerpunkts des Erde-Mond-Systems vom Erdmittelpunkt
dl += + 6.45*sin(D) - 0.42*sin(D-A) + 0.18*sin(D+A)
     + 0.17*sin(D-M3) - 0.06*sin(D+M3);
dr += + 30.76*cos(D) - 3.06*cos(D-A) + 0.85*cos(D+A)
     - 0.58*cos(D+M3) + 0.57*cos(D-M3);
db += + 0.576*sin(U);

// Langperiodische Stoerungen
dl += + 6.40 * sin ( pi2*(0.6983 + 0.0561*T) )
     + 1.87 * sin ( pi2*(0.5764 + 0.4174*T) )
     + 0.27 * sin ( pi2*(0.4189 + 0.3306*T) )
     + 0.20 * sin ( pi2*(0.3581 + 2.4814*T) );

// Ekliptikale Koordinaten ([rad],[AE])
l = pi2 * Frac ( 0.7859453 + M3/pi2 +
                  ((6191.2+1.1*T)*T + dl) / 1296.0e3 );
r = 1.0001398 - 0.0000007 * T + dr * 1.0e-6;
b = db / Arcs;

return Vec3D ( Polar(l,b,r) ); // Ort
}

```

Der Umfang der Funktion SunPos erklärt sich daraus, daß die Relativbewegung von Sonne und Erde bei der geforderten Genauigkeit nicht durch eine einfache Ellipsenbahn beschrieben werden kann. Neben den Störungen der übrigen Planeten – allen voran Venus und Jupiter – wird deshalb auch die monatliche Schwankung des Erdmittelpunktes um den Schwerpunkt des Erde-Mond-Systems berücksichtigt. Strenggenommen bewegt sich ja nicht die Erde selbst, sondern dieser Schwerpunkt in der Ekliptik um die Sonne. Der Umlauf des Mondes um die Erde spiegelt sich dadurch in einer kleinen periodischen Störung der geozentrischen Sonnenbahn wieder. Auf die Einzelheiten der Funktion und ihre Grundlagen soll hier nicht eingegangen werden. Beides wird ausführlich in Kap. 6 dargestellt, wo entsprechende Funktionen für alle Planeten vorgestellt werden.

2.6 Das Programm COCO

Die bisher vorgestellten Funktionen werden nun zu einem vollständigen Programm zusammengefaßt. Im folgenden sind alle relevanten Routinen sowie das Haupt-

programm abgedruckt. Die bereits bekannten Funktionen sind in entsprechenden Bibliotheksmodulen zusammengefaßt, deren zugehörige Schnittstellendateien zu Programmbeginn eingebunden werden.

Die Kernfunktionen des Programms sind in einer Klasse `Position` zusammengefaßt, die neben der Ein- und Ausgabe von Positionen über drei Methoden zur Wahl des Ursprungs (heliozentrisch oder geozentrisch), des Äquinoktiums und des Bezugssystems (ekliptikal oder äquatorial) verfügt. Das Hauptprogramm beschränkt sich damit auf die Deklaration eines Objektes dieser Klasse und den benutzergesteuerten Aufruf seiner Methoden. Auch wenn die Klasse `Position` außerhalb des Coco Programmes keine unmittelbare Anwendung hat, unterstützt der objektorientierte Zugang an dieser Stelle eine kompakte und übersichtliche Programmierung.

```
-----
// Datei: Coco.cpp
// Zweck: Koordinatentransformationen
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
-----

#include <iomanip>
#include <iostream>
#include "APC_Const.h"
#include "APC_Math.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"

// Definition der Klasse "Position"

enum enOrigin { Heliocentric, Geocentric }; // Koordinatenursprung
enum enRefSys { Ecliptic, Equator }; // Bezugssystem
class Position
{
public:
    void Input();
    void SetOrigin(enOrigin Origin);
    void SetRefSys(enRefSys RefSys);
    void SetEquinox(double T_Equinox);
    void Print();
private:
    Vec3D    m_R;          // Koordinatenvektor
    enOrigin m_Origin;     // Koordinatenursprung
    enRefSys m_RefSys;    // Bezugssystem
    double   m_TEquinox;  // Äquinoktium (Julianische Jahrdte seit J2000)
    double   m_MjdEpoch;  // Epoche (Modifiziertes Julianisches Datum)
};

// Datenein- und ausgabe

void Position::Input() { ... }
void Position::Print() { ... }
```

```

//  

// Wechsel des Koordinatenursprungs  

//  

void Position::SetOrigin(enOrigin Origin)  

{  

    double T_Epoch;  

    Vec3D R_Sun;  

    if (Origin!=m_Origin) {  

        // Geozentrische Koordinaten der Sonne bezogen auf das  

        // gegebene Koordinatensystem und Aequinoktium  

        T_Epoch = (m_MjdEpoch-MJD_J2000)/36525.0;  

        if (m_RefSys==Ecliptic)  

            R_Sun = PrecMatrix_Ecl(T_Epoch,m_TEquinox) * SunPos(T_Epoch);  

        else  

            R_Sun = Ecl2EquMatrix(m_TEquinox) *  

                    PrecMatrix_Ecl(T_Epoch,m_TEquinox) * SunPos(T_Epoch);  

        // Wechsel des Koordinatenursprungs  

        if (m_Origin==Heliocentric) {  

            m_R += R_Sun; m_Origin = Geocentric;  

        }  

        else {  

            m_R -= R_Sun; m_Origin = Heliocentric;  

        };  

    };  

}  

//  

// Wechsel des Bezugssystems  

//  

void Position::SetRefSys(enRefSys RefSys)  

{  

    if (RefSys!=m_RefSys) {  

        if (m_RefSys==Equator) {  

            m_R = Equ2EclMatrix(m_TEquinox) * m_R; m_RefSys = Ecliptic;  

        }  

        else {  

            m_R = Ecl2EquMatrix(m_TEquinox) * m_R; m_RefSys = Equator;  

        };  

    };  

}  

//  

// Wechsel des Aequinoktiums  

//  

void Position::SetEquinox(double T_Equinox)  

{  

    if (T_Equinox!=m_TEquinox) {  

        if (m_RefSys==Equator)  

            m_R = PrecMatrix_Equ(m_TEquinox,T_Equinox) * m_R;  

        else  

            m_R = PrecMatrix_Ecl(m_TEquinox,T_Equinox) * m_R;  

        m_TEquinox = T_Equinox;  

    };  

}

```

```

//-----
//  

// Hauptprogramm  

//  

//-----  

void main() {  

    // Variablen  

    Position Pos;  

    char c;  

    bool End = false;  

    double Year;  

    // Vorspann  

    cout << endl  

        << "          COCO: Koordinatentransformationen      " << endl  

        << "          (c) 1999 Oliver Montenbruck, Thomas Pfleger" << endl  

        << endl;  

    // Initialisierung  

    Pos.Input();  

    Pos.Print();  

    // Kommando-Schleife  

    do {  

        // Eingabe eines Kommandos  

        cout << "Bitte Kommando eingeben (?=Hilfe) ... "  

        cin >> c; cin.ignore(81,'\'n\'); c=tolower(c);  

        // Actions  

        switch (c) {  

            case 's':  

                End = true; break;  

            case 'a':  

                Pos.SetRefSys(Equator); Pos.Print(); break;  

            case 'e':  

                Pos.SetRefSys(Ecliptic); Pos.Print(); break;  

            case 'g':  

                Pos.SetOrigin(Geocentric); Pos.Print(); break;  

            case 'h':  

                Pos.SetOrigin(Heliocentric); Pos.Print(); break;  

            case 'n':  

                Pos.Input(); Pos.Print(); break;  

            case 'p':  

                cout << "Neues Aequinoktium (JJJJ.J)      ... "  

                cin >> Year; cin.ignore(81,'\'n\');  

                Pos.SetEquinox((Year-2000.0)/100.0);  

                Pos.Print();  

                break;  

            default:  

                // Hilfetext anzeigen  

                cout << endl  

                    << "Verfuegbare Kommandos" << endl  

                    << " e=ekliptikal, a=aequatorial, p=Praezession, " << endl  

                    << " g=geozentrisch, h=heliozentrisch, n=newe Eingabe," << endl
    }
}

```

```

    << " s=STOP
    << endl;
break;
}
}
while (!End);
cout << endl;
}

```

Das im folgenden angegebene Beispiel soll die Bedienung und die Möglichkeiten von Coco veranschaulichen. Die umzurechnenden Koordinaten können sowohl im äquatorialen als auch im ekliptikalnen System angegeben werden. Dabei kann man jeweils zwischen der Darstellung in kartesischen Koordinaten oder Polarkoordinaten wählen. Diese Auswahl fordert Coco als erste Eingabe an (alle Eingaben sind durch kursive Schrift gekennzeichnet). Wir wählen äquatoriale Koordinaten in polarer Darstellung und geben die Koordinaten des Frühlingspunktes bezogen auf das Äquinoktium 1950 ein. Die Entfernung ist willkürlich zu 1 AE gewählt. Sie muß stets größer als Null sein.

```

COCO: Koordinatentransformationen
(c) 1999 Oliver Montenbruck, Thomas Pfleger

```

Neue Eingabe:

```

Bezugssystem (e=Ekliptik,a=Aequator)      ... a
Format (k=kartesisch, p=polar)             ... p
Koordinaten (RA [h m s] Dec [o ' "] R)   ... 0 0 0.0 0 0 0.0 1.0
Aequinoktium (JJJJ.J)                      ... 1950.0
Ursprung (h=heliozentrisch,g=geozentrisch) ... g
Epoche (JJJJ MM TT HH.H)                   ... 1989 1 1 0.0

```

Coco quittiert die Eingabe, indem es diese Daten in kartesischen Koordinaten und in Polarkoordinaten anzeigt:

```

Geozentrische aequatoriale Koordinaten
(Aequinoktium J1950.0, Epoche 1989/01/01 00.0)

```

```
(x,y,z) = ( 1.00000000, 0.00000000, 0.00000000)
```

```

h m s          o ' "
RA = 0 00 00.00 Dec = + 0 00 00.00 R = 1.00000000

```

```
Bitte Kommando eingeben (?=Hilfe) ... ?
```

```

Verfuegbare Kommandos
e=ekliptikal, a=aequatorial, p=Praezession,
g=geozentrisch, h=heliozentrisch, n=neue Eingabe,
s=STOP

```

```
Bitte Kommando eingeben (?=Hilfe) ...
```

Anschließend wartet das Programm auf die Eingabe eines Kommandos zur Umrechnung der Koordinaten. Mögliche Eingaben sind:

- a** Umwandlung in äquatoriale Koordinaten,
- e** Umwandlung in ekliptikale Koordinaten,
- p** Präzession (Wechsel des Äquinoktiums),
- g** Umwandlung in geozentrische Koordinaten,
- h** Umwandlung in heliozentrische Koordinaten,
- n** Neue Eingabe,
- ?** Hilfe,
- s** STOP (Programmende) .

Wir wollen die eingegebene Position zunächst auf das Äquinoktium 2000 umrechnen und wählen daher die Option p für die Berechnung der Präzession. Nach der Eingabe des Jahres erhalten wir die Position des Frühlingspunkts des Jahres 1950 bezogen auf das neue Äquinoktium 2000. Die Präzession beträgt dabei über ein halbes Grad.

```
Bitte Kommando eingeben (?=Hilfe) ... p
Neues Äquinoktium (JJJJ.J) ... 2000.0
```

```
Geozentrische aequatoriale Koordinaten
(Aquinoktium J2000.0, Epoche 1989/01/01 00.0)
```

```
(x,y,z) = ( 0.99992571, 0.01117889, 0.00485898)
```

```
h m s          o ' "
RA = 0 02 33.73 Dec = + 0 16 42.2 R = 1.00000000
```

Jetzt sollen die im äquatorialen System gegebenen Koordinaten in ekliptikale Koordinaten umgerechnet werden. Dazu wählen wir die Option e und erhalten die umgerechneten Werte wiederum in kartesischer und polarer Darstellung. An die Stelle von Rektaszension RA und Deklination Dec treten nun die ekliptikale Länge L und Breite B. Die Entfernung bleibt wie auch beim vorhergehenden Rechenschritt unverändert.

```
Bitte Kommando eingeben (?=Hilfe) ... e
```

```
Geozentrische ekliptikale Koordinaten
(Aquinoktium J2000.0, Epoche 1989/01/01 00.0)
```

```
(x,y,z) = ( 0.99992571, 0.01218922, 0.00001132)
```

```
o ' "          o ' "
L = 0 41 54.27 B = + 0 00 02.3 R = 1.00000000
```

Mit der Option h ist es nun möglich, die gegebene Position von geozentrischen Koordinaten in heliozentrische umzuwandeln. Als Epoche dient dabei der eingangs angegebene Zeitpunkt. Da wir beim letzten Dialogschritt die Umrechnung in ekliptikale Koordinaten gewählt hatten, erhalten wir nun heliozentrische, ekliptikale Koordinaten. Man beachte, daß sich die Entfernung dabei verändert.

```
Bitte Kommando eingeben (?=Hilfe) ... h
Heliozentrische ekliptikale Koordinaten
(Aequinoktium J2000.0, Epoche 1989/01/01 00.0)
(x,y,z) = ( 0.81725247, 0.97838164, 0.00003597)

o ' "
L = 50 07 39.50 B = + 0 00 05.8 R = 1.27480674
```

Die Umrechnung in ekliptikale Koordinaten kann mit der Option a wieder rückgängig gemacht werden, die eine Umrechnung der aktuellen (ekliptikalnen) Koordinaten in äquatoriale Koordinaten bewirkt.

```
Bitte Kommando eingeben (?=Hilfe) ... a
Heliozentrische aequatoriale Koordinaten
(Aequinoktium J2000.0, Epoche 1989/01/01 00.0)
(x,y,z) = ( 0.81725247, 0.89763329, 0.38921086)

h m s          o ' "
RA = 3 10 44.07 Dec = +17 46 36.5 R = 1.27480674
```

Nun rechnen wir die aktuellen Koordinaten wieder auf das Äquinoktium 1950 zurück.

```
Bitte Kommando eingeben (?=Hilfe) ... p
Neues Aequinoktium (JJJJ.J) ... 1950.0
Heliozentrische aequatoriale Koordinaten
(Aequinoktium J1950.0, Epoche 1989/01/01 00.0)
(x,y,z) = ( 0.82911747, 0.88843066, 0.38521087)

h m s          o ' "
RA = 3 07 54.68 Dec = +17 35 17.2 R = 1.27480674
```

Die Transformation dieser Werte in geozentrische Koordinaten müßte nun wieder die ursprünglichen Koordinaten des Frühlingspunktes liefern, da wir dann zu den ersten drei Umrechnungen auch die inversen Transformationen aufgerufen haben. Die unvermeidlichen Rundungsfehler des Rechners verhindern allerdings in manchen Fällen, daß exakt dieselben Koordinaten wie zu Beginn unserer Beispiellechnung ausgegeben werden.

```
Bitte Kommando eingeben (?=Hilfe) ... g
Geozentrische aequatoriale Koordinaten
(Aequinoktium J1950.0, Epoche 1989/01/01 00.0)
(x,y,z) = ( 1.00000000, -0.00000000, 0.00000000)

h m s          o ' "
RA = 24 00 00.00 Dec = + 0 00 00.0 R = 1.00000000
```

Die Eingabe von **s** beendet den Lauf des Programms.

Bitte Kommando eingeben (?=Hilfe) ... s

Die hier angegebenen Zahlenwerte können je nach verwendetem C++-Compiler in geringem Maße differieren.

3. Auf- und Untergangsrechnung

3.1 Das Horizontsystem des Beobachters

Die bisher behandelten ekliptikalnen und äquatorialen Koordinaten sind durch die mittlere Ebene der Erdbahn und die Lage der Erdachse besonders ausgezeichnet. Für einen Beobachter auf der Erdoberfläche ist jedoch zunächst keines dieser beiden raumfesten Systeme besonders dienlich. Da er an der täglichen Drehung der Erde teilnimmt, ohne es bewußt zu merken, hat er den Eindruck, als würden sich Sonne, Mond und Sterne auf großen Bögen im Laufe eines Tages von Osten nach Westen bewegen und dabei im Meridian ihre höchste Stellung über dem Horizont erreichen.

Der Verlauf der scheinbaren Sternbahnen hängt von der geographischen Breite des Beobachtungsortes ab. Auf der Südhalbkugel der Erde erreichen die meisten Gestirne ihre größte Höhe über dem Horizont nicht im Süden, sondern im Norden. Entsprechend ist die Orientierung am Sternhimmel für einen Beobachter, der den Himmelsanblick gemäßigter nördlicher Breiten gewöhnt ist, erschwert. Für ihn scheinen die bekannten Sternbilder nämlich auf dem Kopf zu stehen.

Zwei Punkte an der Himmelssphäre sind besonders ausgezeichnet: der Zenit (das ist der Punkt senkrecht über dem Beobachter) und der Himmelsnordpol (Abb. 3.1). Darunter versteht man denjenigen Punkt der scheinbaren Himmelskugel, der in Richtung der Erdachse weist und um den sich deswegen alle Gestirne auf konzentrischen Kreisbahnen zu bewegen scheinen. Die Höhe dieses Punktes über dem Horizont entspricht der geographischen Breite φ des Beobachters. Legt man durch den Pol und den Zenit einen Großkreis, so schneidet dieser sogenannte *Meridian* den Horizont genau in den Himmelsrichtungen Nord und Süd.

Als Koordinaten im Horizontsystem verwendet man das Azimut (A) und die Höhe (h), die sich etwa mit Hilfe eines Theodoliten bestimmen lassen (vgl. Abb. 3.1). Die Höhe über dem Horizont gibt den Winkel an, um den man den Theodoliten aus der Waagerechten nach oben neigen muß. Das Azimut beschreibt, um welchen Winkel der Theodolit um seine senkrechte Achse gedreht werden muß, um einen gesuchten Punkt von der Südrichtung ausgehend einzustellen. Gemäß dieser Festlegung ergibt sich für das Azimut des Westpunkts ein Wert von $A = 90^\circ$ und entsprechend für den Ostpunkt ein Azimut von $A = 270^\circ$ (oder $A = -90^\circ$). Leider ist neben dieser Definition noch eine abweichende, vor allem in der Navigation verwendete Zählung gebräuchlich. Bei ihr bildet der Nordpunkt den Ursprung des Azimuts. Nach dieser Vereinbarung hat dann der Ostpunkt ein Azimut von 90° . Man sollte sich im Zweifelsfalle immer vergewissern, nach welcher dieser beiden Definitionen ein Azimut angegeben ist!

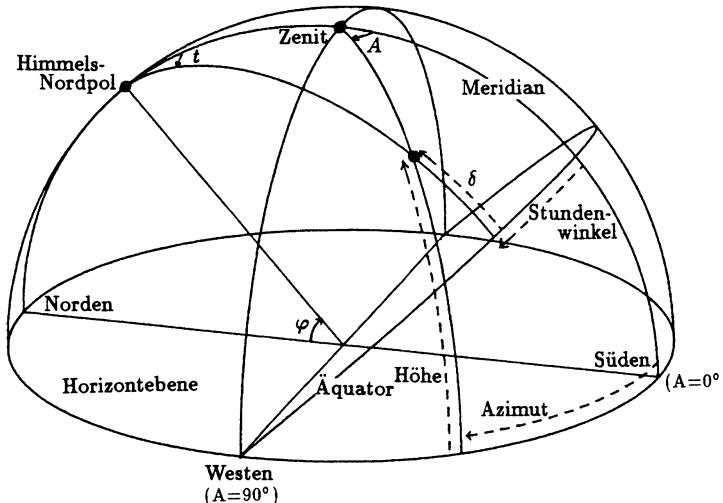


Abb. 3.1. Das Horizontsystem

Aufgrund der Erddrehung ändern sich Azimut und Höhe eines Gestirns mit gegebener Rektaszension und Deklination ständig. Richtet man ein Fernrohr im Horizontsystem fest aus, so wandern im Verlauf der Zeit aber nur solche Sterne durch das Blickfeld, die die gleiche Deklination besitzen und sich nur in ihrer Rektaszension unterscheiden. Die Bestimmung der Deklination eines Gestirns aus Azimut und Höhe ist daher von der Zeit unabhängig. Im Fall der Rektaszension lässt sich dagegen nur die Differenz gegenüber Sternen im Meridian bestimmen.

Für die gegenseitige Umrechnung der horizontalen und äquatorialen Koordinaten führt man den *Stundenwinkel* τ ein. Dieser gibt die Differenz zwischen der Rektaszension des betrachteten Sterns und der Rektaszension der Sterne im Meridian an. Der Stundenwinkel τ wird wie die Rektaszension α meist im Stundenmaß ($1^h \equiv 15^\circ$) gemessen und entspricht dann ungefähr¹ der Uhrzeit, die seit dem letzten Meridiandurchgang des Sterns vergangen ist. A und τ haben mit den hier getroffenen Vereinbarungen gleiches Vorzeichen. Mit den kartesischen Koordinaten

$$\mathbf{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos h \cos A \\ \cos h \sin A \\ \sin h \end{pmatrix} \quad \hat{\mathbf{r}} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = \begin{pmatrix} \cos \delta \cos \tau \\ \cos \delta \sin \tau \\ \sin \delta \end{pmatrix} \quad (3.1)$$

gelten die Umrechnungen

$$\begin{aligned} x &= +\hat{x} \sin \varphi - \hat{z} \cos \varphi & \hat{x} &= +x \sin \varphi + z \cos \varphi \\ y &= +\hat{y} & \hat{y} &= +y \\ z &= +\hat{x} \cos \varphi + \hat{z} \sin \varphi & \hat{z} &= -x \cos \varphi + z \sin \varphi \end{aligned} \quad (3.2)$$

oder kurz $\mathbf{r} = \mathbf{R}_y(\pi/2 - \varphi)\hat{\mathbf{r}}$. Darin ist φ die geographische Breite des Beobachters, kennzeichnet die äquatorialen Koordinaten. Durch Einsetzen erhält man daraus

¹Eine Erdumdrehung dauert nur $23^h 56^m$.

schließlich das Gleichungssystem

$$\begin{aligned}\cos h \cos A &= +\cos \delta \cos \tau \sin \varphi - \sin \delta \cos \varphi \\ \cos h \sin A &= +\cos \delta \sin \tau \\ \sin h &= +\cos \delta \cos \tau \cos \varphi + \sin \delta \sin \varphi\end{aligned}\quad (3.3)$$

zur Berechnung von Azimut und Höhe aus Stundenwinkel und Deklination. Umgekehrt liefert

$$\begin{aligned}\cos \delta \cos \tau &= +\cos h \cos A \sin \varphi + \sin h \cos \varphi \\ \cos \delta \sin \tau &= +\cos h \sin A \\ \sin \delta &= -\cos h \cos A \cos \varphi + \sin h \sin \varphi\end{aligned}\quad (3.4)$$

den Stundenwinkel und die Deklination aus den vorgegebenen Horizontkoordinaten. Zur Umrechnung zwischen dem äquatorialen System und dem System der Horizontkoordinaten eignen sich die beiden Funktionen Equ2Hor und Hor2Equ, die hier der Vollständigkeit wegen angegeben sind.

```
//-----
// Equ2Hor: Transformation aequatorialer Koordinaten ins Horizontsystem
// Dec      Deklination [rad]
// tau      Stundenwinkel [rad]
// lat      Geographische Breite des Beobachters [rad]
// h       Hoehe [rad]
// Az      Azimut [rad]
//-----
void Equ2Hor ( double Dec, double tau, double lat,
               double& h, double& Az )
{
    Vec3D e_equ, e_hor;
    e_equ = Vec3D(Polar(tau,Dec));           // Einheitsvektor im Horizontsystem
    e_hor = R_y(pi/2.0-lat) * e_equ;         // Einheitsvektor im aequator. System
    Az = e_hor[phi];                         // Polarwinkel
    h  = e_hor[theta];
}

//-----
// Hor2Equ: Transformation vom Horizontsystem in aequatoriale Koordinaten
// h       Hoehe [rad]
// Az      Azimut [rad]
// lat    Geographische Breite des Beobachters [rad]
// Dec    Deklination [rad]
// tau    Stundenwinkel [rad]
//-----
void Hor2Equ ( double h, double Az, double lat,
                double& Dec, double& tau )
{
    Vec3D e_equ, e_hor;
    e_hor = Vec3D(Polar(Az,h));             // Einheitsvektor im Horizontsystem
    e_equ = R_y(-(pi/2.0-lat)) * e_hor;    // Einheitsvektor im aequator. System
    tau = e_equ[phi];                       // Polarwinkel
    Dec = e_equ[theta];
}
```

Für die Bestimmung von Auf- und Untergangszeiten, die das Ziel dieses Kapitels ist, benötigt man nur die Gleichung

$$\sin h = \cos \varphi \cos \delta \cos \tau + \sin \varphi \sin \delta \quad . \quad (3.5)$$

Sie gestattet die Berechnung der Höhe h aus gegebenen Werten für die geographische Breite φ , die Deklination δ und den Stundenwinkel τ .

Bevor wir diese Beziehung verwenden können, ist jedoch noch einige Vorbereitung nötig. Zunächst fehlt uns die Möglichkeit, die Koordinaten (α, δ) von Sonne und Mond zu bestimmen. Anschließend ist die Frage zu klären, wie wir zu einer gegebenen Zeit den Stundenwinkel aus der bekannten Rektaszension berechnen können. Zuletzt müssen wir uns noch mit einer Reihe von Korrekturen auseinandersetzen, die die beobachtete Horizonthöhe beeinflussen. Erst danach werden wir wieder auf die obige Gleichung zurückkommen.

3.2 Sonne und Mond

Bei der Berechnung von Auf- und Untergängen werden keine hohen Ansprüche an die Genauigkeit der Sonnen- und Mondkoordinaten gestellt. Die Funktionen `MiniSun` und `MiniMoon` enthalten deshalb nur die wichtigsten Terme zur Beschreibung der entsprechenden Bahnen. Es handelt sich dabei um stark verkürzte Versionen der Funktionen `SunPos` und `MoonPos`, die in Kap. 6 und Kap. 8 ausführlich besprochen werden. Die Umrechnung der ekliptikalalen Länge und Breite in äquatoriale Koordinaten wurde mit eingefügt, so daß beide Funktionen ohne weitere Operationen Rektaszension und Deklination des jeweiligen Objektes liefern.

```
-----
// MiniMoon: Berechnet Rektaszension und Deklination des Mondes mit einer
// analytischen Reihe geringer Genauigkeit
// T          Zeit in Julianischen Jahrhunderten seit J2000
// RA         Rektaszension des Mondes in [rad]
// Dec        Deklination des Mondes in [rad]
-----
void MiniMoon (double T, double& RA, double& Dec)
{
    const double eps = 23.43929111*Rad;
    double L_0, l_ls, F, D, dL, S, h, N;
    double l_Moon, b_Moon;
    Vec3D e_Moon;
    // Mittlere Elemente der Mondbahn
    L_0 = Frac (0.606433 + 1336.855225*T);           // mittlere Laenge [Umlaeufe]
    l = pi2*Frac (0.374897 + 1325.552410*T);        // Mittlere Anomalie des Mondes
    ls = pi2*Frac (0.993133 + 99.997361*T);         // Mittlere Anomalie der Sonne
    D = pi2*Frac (0.827361 + 1236.853086*T);        // Laengendiff. Mond-Sonne
    F = pi2*Frac (0.259086 + 1342.227825*T);        // Knotenabstand
    // Störungen in Laenge und Breite
    dL = +22640*sin(l) - 4586*sin(l-2*D) + 2370*sin(2*D) + 769*sin(2*l)
        - 668*sin(ls) - 412*sin(2*F) - 212*sin(2*l-2*D) - 206*sin(l+ls-2*D)
        + 192*sin(l+2*D) - 165*sin(ls-2*D) - 125*sin(D) - 110*sin(l+ls)
        + 148*sin(l-ls) - 55*sin(2*F-2*D);
    S = F + (dL+412*sin(2*F)+541*sin(ls)) / Arcs;
```

```

h = F-2*D;
N = -526*sin(h) + 44*sin(l+h) - 31*sin(-l+h) - 23*sin(ls+h)
    + 11*sin(-ls+h) - 25*sin(-2*l+F) + 21*sin(-l+F);
// Ekliptikale Laenge und Breite
l_Moon = pi2 * Frac( L_0 + dL/1296.0e3 ); // [rad]
b_Moon = ( 18520.0*sin(S) + N ) / Arcs; // [rad]
// Aequatoriale Koordinaten
e_Moon = R_x(-eps) * Vec3D(Polar(l_Moon,b_Moon));
RA = e_Moon[phi];
Dec = e_Moon[theta];
}

//-----
// MiniSun: Berechnet Rektaszension und Deklination der Sonne unter Verwendung
// einer analytischen Reihenentwicklung mit geringer Genauigkeit
// T Zeit in Julianischen Jahrhunderten seit J2000
// RA Rektaszension der Sonne in [rad]
// Dec Deklination der Sonne in [rad]
//-----
void MiniSun (double T, double& RA, double& Dec)
{
    const double eps = 23.43929111*Rad;
    double L,M;
    Vec3D e_Sun;
    // Mittlere Anomalie und ekliptikale Laenge
    M = pi2 * Frac ( 0.993133 + 99.997361*T );
    L = pi2 * Frac ( 0.7859453 + M/pi2 +
        (6893.0*sin(M)+72.0*sin(2.0*M)+6191.2*T) / 1296.0e3 );
    // Aequatoriale Position
    e_Sun = R_x(-eps) * Vec3D(Polar(L,0.0));
    RA = e_Sun[phi];
    Dec = e_Sun[theta];
}

```

3.3 Sternzeit und Stundenwinkel

Die Rektaszension von Sonne und Mond genügt alleine noch nicht, um deren Höhe über dem Horizont zu berechnen. Wir müssen zusätzlich die Rektaszension der Sterne im Meridian kennen, und daraus den Stundenwinkel bestimmen. Welche Sterne dies gerade sind, hängt vom jeweiligen Ort und der augenblicklichen Zeit ab. Während sich die Erde im Lauf des Tages um ihre Achse dreht, sieht der Beobachter Sterne aller Rektaszensionen von 0^{h} bis 24^{h} durch den Meridian wandern. In rund einer Stunde ändert sich die Rektaszension der jeweils kulminierenden Sterne um 1^{h} .

Die regelmäßige Bewegung der Sterne eignet sich sehr gut zur Festlegung einer neuen Uhrzeit, die man als *Sternzeit* bezeichnet. Sie ist für einen festen Ort als Rektaszension derjenigen Sterne definiert, die dort gerade im Meridian stehen. Gemäß dieser Definition kann man die Sternzeit unmittelbar durch Beobachtung des Himmels ermitteln.

Die Notwendigkeit, eine Sternzeit als Ergänzung zur Sonnenzeit einzuführen, ergibt sich aus einem kleinen Unterschied zwischen der Länge eines Sonnentages

und der Dauer einer Erdumdrehung. Die Uhren, auf denen wir ständig die Zeit ablesen, sind so geeicht, daß ein Tag in 24^h eingeteilt ist. Ein Tag ist dabei der Wechsel von Hell und Dunkel, der durch den Lauf der Sonne bestimmt ist. 24^h sind gerade die Zeit, die die Sonne im Mittel von einem Meridiandurchgang zum nächsten braucht. Mißt man dagegen die entsprechende Zeitspanne für einen Stern, so stellt man fest, daß dieser dazu nur 23^h56^m4^s091 benötigt. Diese kürzere Zeitdauer, die man im Gegensatz zum Sonnentag als Sternzeit bezeichnet, hat genau die Länge einer Erdumdrehung. Die Ursache für die Differenz von rund 4^m liegt im jährlichen Umlauf der Erde um die Sonne begründet. Die Sonne ändert durch diese Bewegung ihre Rektaszension um 360° ≈ 24^h pro Jahr, also täglich um rund 4^m. Dadurch ist die Zeit zwischen zwei Meridiandurchgängen für die Sonne um eben diesen Betrag größer als für einen Stern.

Zu gegebener Weltzeit UT kann man zunächst die Sternzeit von Greenwich über die Beziehung

$$\begin{aligned}\Theta_0 &= 24110^s 54841 + 8640184^s 812866 \cdot T_0 + 1.0027379093 \cdot UT \\ &\quad + 0^s 093104 \cdot T^2 - 0^s 0000062 \cdot T^3\end{aligned}\quad (3.6)$$

mit

$$T_0 = \frac{JD_0 - 2451545}{36525} \quad \text{und} \quad T = \frac{JD - 2451545}{36525}$$

berechnen. Dabei sind JD und JD₀ das Julianische Datum zum Beobachtungszeitpunkt und das Julianische Datum um 0^h UT des Beobachtungsdatums.

```
//-----
// GMST: Mittlere Greenwich-Sternzeit
//   MJD      Zeit als Modifiziertes Julianisches Datum
//   <return>: Mittlere Greenwich-Sternzeit in [rad]
//-----
double GMST (double MJD)
{
    const double Secs = 86400.0;           // Anzahl der Sekunden je Tag
    double MJD_0, UT, T_0, T, gmst;
    MJD_0 = floor (MJD);
    UT    = Secs*(MJD-MJD_0);           // [s]
    T_0   = (MJD_0-51544.5)/36525.0;
    T     = (MJD - 51544.5)/36525.0;
    gmst = 24110.54841 + 8640184.812866*T_0 + 1.0027379093*UT
          + (0.093104-6.2e-6*T)*T*T;        // [sec]
    return (pi2/Secs)*Modulo(gmst,Secs); // [Rad]
```

Für einen Ort mit der geographischen Länge λ unterscheidet sich die Sternzeit um $\lambda/15^\circ$ Stunden von der Greenwich-Sternzeit:

$$\Theta = \Theta_0 + \lambda \cdot 1^h / 15^\circ . \quad (3.7)$$

λ wird hier *positiv nach Osten* gezählt (für München ergibt sich beispielsweise ein Wert von $\lambda = +11^\circ 6'$). Der Stundenwinkel τ hat damit für einen Stern der Rektaszension α den Wert

$$\tau = \Theta - \alpha . \quad (3.8)$$

Als Beispiel für die Umrechnung von Rektaszension und Stundenwinkel mit Hilfe der Sternzeit werden im folgenden Programmbeispiel Azimut und Höhe des Sterns Deneb (α Cyg) für einen gegebenen Beobachtungsort und -zeitpunkt bestimmt:

```

double lambda = +11.6*Rad;                                // Beobachtungsort Muenchen
double phi    = 48.1*Rad;
double RA     = Ddd(20,41,12.8)*Rad                      // Aequ. Koord. Deneb
double Dec    = Ddd(45,15,25.8)*Rad
double ModJD = Mjd ( 1993,8,1, 21,0,0.0 );   // 1. Aug. 1993, 21:00 UT

double tau,h,Az;

tau = GMST(ModJD) + lambda - RA;                         // Stundenwinkel [rad]
Equ2Hor ( Dec,tau,phi, h,Az );                            // Azimut und Hoehe

cout << "Deneb:" << endl                               // Ausgabe
<< setprecision(2)
<< "Hoehe =" << setw(7) << h << " Grad"
<< "Azimut =" << setw(7) << Az << " Grad";

```

3.4 Weltzeit und Ephemeridenzeit

Wir haben nun schon häufig den Begriff „Zeit“ benutzt, ohne uns darüber nähere Gedanken zu machen. In der Astronomie begegnet man jedoch einer ganzen Reihe von Zeitzählungen, die nebeneinander verwendet werden. Sie lassen sich im wesentlichen in zwei Klassen einteilen, deren wichtigste Vertreter die *dynamische Zeit* (TDB/TDT) und die *Weltzeit* (UT) sind. Die unterschiedlichen Zielvorstellungen, die diesen Zeiten zugrunde liegen, sollen hier etwas ausführlicher erläutert werden.

Ihrem Konzept nach ist die dynamische Zeit eine Zeit, die die Grundlage für die Beschreibung astronomischer Vorgänge im Rahmen der Physik bildet. Eine Sekunde der dynamischen Zeit ist über die Periodenzahl eines bestimmten Übergangs im Caesium-Atom definiert und wird heute dementsprechend mit Atomuhren gemessen. Die dynamische Zeit ersetzt seit 1984 die *Ephemeridenzeit* (ET). Die Grundlage dieser früheren Festlegung waren astronomische Ephemeriden, also Tafeln der Bewegung von Sonne, Mond und Planeten, die nach den Gesetzen der klassischen Mechanik berechnet waren. Die Ephemeridenzeit konnte durch Vergleich von beobachteten Positionen dieser Himmelskörper mit den vorausberechneten Ephemeriden bestimmt werden. Der Grund für die Einführung der dynamischen Zeit liegt in der Erkenntnis der Relativitätstheorie, daß es keine universelle Zeit gibt, sondern daß der Ablauf der Zeit auch vom Ort und der Bewegung eines Bezugssystems abhängt, in dem die Zeit gemessen wird. Dies bedingt insbesonders die prinzipielle Unterscheidung zwischen der terrestrischen (auf die Erde bezogenen) dynamischen Zeit TDT und der TDB, die sich auf den Schwerpunkt des Sonnensystems bezieht. Für unsere Zwecke können aber alle drei Zeiten ET, TDB und TDT ohne Bedenken gleichgesetzt werden.

Im Gegensatz zur Ephemeridenzeit und zur dynamischen Zeit ist die *Weltzeit* (UT) eine *ungleichförmige* Zeitskala. Die UT ist heute die beste Verwirklichung einer Sonnenzeit. Mit ihrer Einführung versucht man zu erreichen, daß ein Tag

auch über einige tausend Jahre hinweg im Mittel 24^h lang ist. Dies führt aber dazu, daß die Länge einer Sekunde Weltzeit nicht konstant ist, weil die tatsächliche mittlere Tageslänge von der Drehung der Erde und der scheinbaren Bewegung der Sonne (also der Jahreslänge) abhängt. Leider ist es nicht möglich, die Weltzeit durch Umrechnung aus der dynamischen Zeit zu bestimmen, weil die Rotation der Erde nicht genau vorhergesagt werden kann. Jede Veränderung der Erddrehung ändert aber die Tageslänge und muß daher auch in der UT berücksichtigt werden. Man definiert deshalb die Weltzeit als Funktion der Sternzeit, die ja direkt die Drehung der Erde wiederspiegelt. 0^h Weltzeit eines Tages ist als der Augenblick definiert, in dem die Sternzeit von Greenwich (GMST) den festgelegten Wert

$$\begin{aligned} \text{GMST}(0^{\text{h}}\text{UT}) = & 24110^{\text{s}}.54841 + 8640184^{\text{s}}.812866 \cdot T_0 \\ & + 0^{\text{s}}.093104 \cdot T_0^2 - 0^{\text{s}}.0000062 \cdot T_0^3 \end{aligned}$$

mit

$$T_0 = \frac{\text{JD}(0^{\text{h}}\text{UT}) - 2451545}{36525}$$

hat. Diese Gleichung ist uns bereits in Abschn. 3.3 begegnet, wo wir sie zur Berechnung der Sternzeit zu gegebener Weltzeit benutzt haben. Man halte sich aber vor Augen, daß die Sternzeit die eigentliche Beobachtungsgröße ist, aus der dann die Weltzeit abgeleitet wird.

Die Differenz zwischen der Weltzeit und der dynamischen Zeit (beziehungsweise der Ephemeridenzeit) läßt sich nur nachträglich bestimmen. Die Tabelle 3.1 gibt eine Übersicht über die Werte $\Delta T = \text{ET}-\text{UT}$ ($\text{ET}=\text{TDB}=\text{TDT}$) im Verlauf dieses Jahrhunderts. Gegenwärtig wächst ΔT um etwa 0.5 bis 1.0 Sekunden pro Jahr.

Tabelle 3.1. Verlauf der Differenz $\Delta T = \text{ET}-\text{UT}$ in s

Jahr	ET-UT	Jahr	ET-UT	Jahr	ET-UT	Jahr	ET-UT
1900	-2.72						
1905	3.86	1930	24.02	1955	31.07	1980	50.54
1910	10.46	1935	23.93	1960	33.15	1985	54.34
1915	17.20	1940	24.33	1965	35.73	1990	56.86
1920	21.16	1945	26.77	1970	40.18	1995	60.82
1925	23.62	1950	29.15	1975	45.48	2000	(65.0)

Die Uhrzeit, die wir im Alltag verwenden, leitet sich von der *koordinierten Weltzeit* (UTC) ab. Die UTC wird von Atomuhren abgelesen, hat also zunächst die gleiche Ganggeschwindigkeit wie die dynamische Zeit. Durch Schaltsekunden, die bis zu zweimal im Jahr eingelegt werden können, wird aber erreicht, daß die UTC nie um mehr als 0.9 Sekunden von der Weltzeit UT abweicht. Die UTC bildet die Grundlage für die Zeitmessung der ganzen Erde. Dazu ist jeder Ort einer Zeitzone zugeordnet, in der sich die offizielle Zeit um jeweils ganze (oder halbe) Stunden von der UTC unterscheidet. In der Praxis hat man die Zeitzonen geographischen Gegebenheiten und insbesonders den Ländergrenzen angepaßt, um

nach Möglichkeit zu verhindern, daß in ein und demselben Land mehrere Zonenzeiten verwendet werden müssen. Welche Zeitzone in einem bestimmten Gebiet verwendet wird, ist nach internationaler Übereinkunft festgelegt und kann einer Zeitzonenkarte entnommen werden.

An dieser Stelle stellt sich natürlich die Frage, mit welcher Zeit wir zu arbeiten haben, wenn wir die Auf- und Untergänge von Sonne und Mond berechnen wollen. Aus den obigen Definitionen ergeben sich ganz allgemein die folgenden Faustregeln für die Verwendung der Weltzeit und der Ephemeridenzeit:

- Die Weltzeit (UT) dient zur Berechnung der Sternzeit.
- Die Ephemeridenzeit (ET) oder die dynamische Zeit (TDB/TDT) dienen zur Berechnung von Sonnen-, Mond- und Planetenephemeren.

Betrachten wir dazu als Beispiel die einzelnen Schritte, die eine Berechnung der Mondhöhe über dem Horizont erfordert. Der gewählte Zeitpunkt sei etwa der 1. Januar 1982 um 0^h Mitteleuropäischer Zeit, der Beobachtungsort sei München ($\varphi = 48^\circ 1'$, $\lambda = +11^\circ 6'$). Der geringfügige Unterschied zwischen UTC und UT soll hier nicht weiter betrachtet werden, so daß wir die Weltzeit aus UT=MEZ-1^h erhalten. Nach den obigen Definitionen müssen wir zur Berechnung der Sternzeit die Weltzeit verwenden, während die Ephemeridenzeit in die Berechnung der Mondkoordinaten eingeht. Dies führt zu der folgenden Umsetzung in ein kleines Programmsegment:

```

double lambda = +11.6*Rad;                                // Beobachtungsort Muenchen
double phi     = 48.1*Rad;

double MjdMEZ = Mjd ( 1982,1,1, 0,0,0.0 );      // 1. Jan. 1982, 00:00 MEZ
double MEZ_UT = 1.0                                     // Zonenzeit [h]
double ET_UT  = 52.17;                                 // [s] fuer 1982

double MjdUT,MjdET, RA,Dec, sinH;

MjdUT  = MjdMEZ - MEZ_UT/24.0;                      // MJD Weltzeit
MjdET  = MjdUT  + ET_UT/86400.0;                    // MJD Ephemeridenzeit

MiniMoon ( MjdET,RA,Dec);                           // Aequatoriale Koord.

tau   = GMST(MjdUT) + lambda - RA;                 // Stundenwinkel [rad]
sinH = sin(phi)*sin(Dec)                            // Sinus der Horizonthöhe
       + cos(phi)*cos(Dec)*cos(tau);
```

Andererseits kann man sich fragen, welchen Fehler man begeht, wenn man den Unterschied zwischen Weltzeit und Ephemeridenzeit vernachlässigt. Berechnet man die Mondkoordinaten, indem man vereinfachend die UT anstelle der ET einsetzt, dann hat das obige Beispiel die folgende Gestalt:

```

MjdUT  = MjdMEZ - MEZ_UT/24.0;                      // MJD Weltzeit
MiniMoon ( MjdUT,RA,Dec);                           // Aequatoriale Koord.
```

```

tau = GMST(MjdUT) + lambda - RA;           // Stundenwinkel [rad]
sinH = sin(phi)*sin(Dec)                   // Sinus der Horizonthoehe
+ cos(phi)*cos(Dec)*cos(tau);

```

Während die Sternzeit also korrekt ausgewertet wird, sind die Mondkoordinaten hier für einen um $\Delta T = ET - UT$ zu *frühen* Zeitpunkt berechnet. Gegenwärtig liegt ΔT bei etwa einer Minute, einem Zeitraum, in dem sich der Mond um rund $30''$ weiterbewegt. Dieser Fehler ist bereits kleiner als der Fehler, den wir durch die vereinfachte Berechnung der Mondkoordinaten in *MiniMoon* begehen. Die Auf- und Untergangszeit wird dadurch nur um rund $3''$ verfälscht. Für die Sonne fallen diese Zahlen noch kleiner aus. Im Programm *Sunset* können wir daher guten Gewissens ΔT in der beschriebenen Weise vernachlässigen. Bei späteren Anwendungen wie der Berechnung genauer Planetenpositionen oder bei der Vorhersage von Sternbedeckungen ist es jedoch notwendig, sorgfältig zwischen den verschiedenen Zeiten zu unterscheiden. Aus diesem Grunde wurde bereits hier ausführlich auf die Besonderheiten der astronomischen Zeitrechnung eingegangen.

3.5 Parallaxe und Refraktion

Bisher haben wir Rektaszension und Deklination eines Himmelskörpers immer in geozentrischen äquatorialen Koordinaten angegeben, also in einem Koordinatensystem, das seinen Ursprung im Erdmittelpunkt hat. Wir befinden uns jedoch nicht in der Mitte der Erde, sondern beobachten von ihrer Oberfläche aus. Welche Abweichungen können sich dadurch zwischen den berechneten und den beobachteten Koordinaten ergeben?

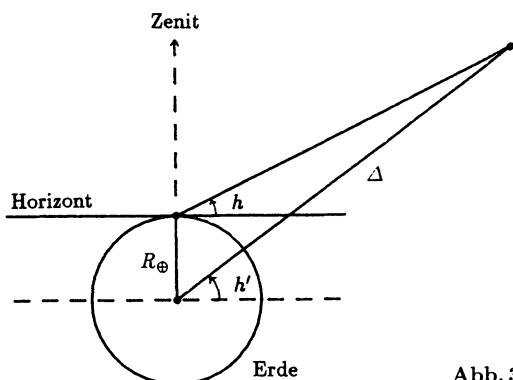


Abb. 3.2. Zur Parallaxe

Bei der Beobachtung der Fixsterne bemerkt man praktisch keine Unterschiede zwischen den geozentrischen und den *topozentrischen* Koordinaten, die man erhält, wenn man durch eine Parallelverschiebung den Nullpunkt des Koordinatensystems vom Erdmittelpunkt zum Beobachter auf die Erdoberfläche verlegt. Ihre Entfernung ist riesig im Verhältnis zu der Distanz des Beobachters vom Erdmittelpunkt. Sobald man aber nähergelegene Himmelskörper wie die Planeten, die Sonne oder den Mond betrachtet, ergeben sich merkliche Differenzen.

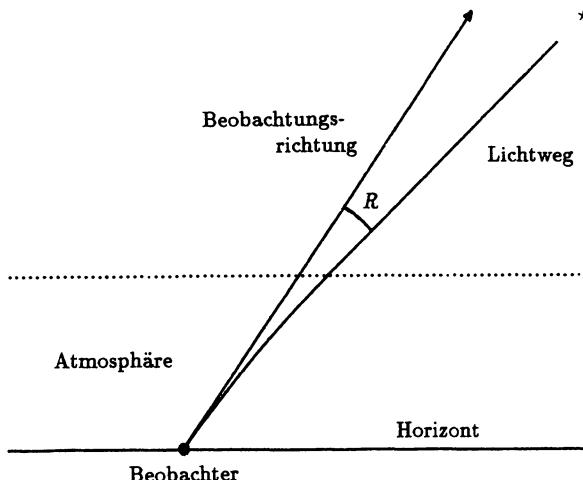


Abb. 3.3. Zur Refraktion

Dieser Unterschied wird *Parallaxe* genannt und bewirkt, daß ein von der Erdoberfläche aus beobachtetes Objekt etwas tiefer steht, als wenn man es vom Erdmittelpunkt aus beobachten könnte. Die topozentrische Höhe h ist geringer als die geozentrische Höhe h' (vgl. Abb. 3.2). Steht das Objekt im Zenit, dann verschwindet die Parallaxe. Mit abnehmender Höhe aber macht sie sich zunehmend bemerkbar und erreicht bei der Höhe $h = 0^\circ$ ihren größten Wert. Dieser wird als *Horizontalparallaxe* bezeichnet und berechnet sich zu

$$\pi = \arcsin(R_\oplus/\Delta) , \quad (3.9)$$

wobei $R_\oplus \approx 6378$ km die Entfernung des Beobachters vom Erdmittelpunkt und Δ die *geozentrische* Entfernung des Objekts angibt. Setzt man die Entfernung der Sonne in (3.9) ein, dann erhält man einen Betrag von $\pi_\odot = 8.^{\hspace{-0.1em}s}8$. Beim Mond als erdnächstem Nachbarn erreicht die mittlere Horizontalparallaxe dagegen den nicht mehr zu vernachlässigenden Betrag von

$$\pi_{\text{Mond}} = \arcsin \left(\frac{6378 \text{ km}}{384400 \text{ km}} \right) \approx 57' .$$

Man kann an dieser Stelle schon erkennen, daß wir für die Auf- und Untergangsrechnung nicht die *geozentrische*, sondern die *topozentrische* Höhe heranziehen müssen. Zuvor soll aber noch auf einen anderen Effekt eingegangen werden.

Beim Passieren der Erdatmosphäre werden die Lichtstrahlen beim Übergang aus dem Vakuum des Weltraums in die optisch dichtere Atmosphäre gemäß dem Brechungsgesetz zum Lot hin abgelenkt (vgl. Abb. 3.3). Diese Erscheinung wird *Refraktion* genannt. Die Sterne erscheinen einem Beobachter am Boden infolgedessen etwas „angehoben“. Ein unter flachem Winkel eintretender Lichtstrahl hat dabei einen längeren Weg durch Luftmassen unterschiedlicher Dichte (und Brechungscharakteristik) zurückzulegen, als ein steil eintretender Lichtstrahl. Daher ist die Refraktion umso stärker, je geringer die Höhe des Objektes ist. Der Maximalwert von rund $34'$ wird am Horizont erreicht. Zur Ergänzung sind in Tabelle

3.2 einige Daten angegeben. R ist zur berechneten topozentrischen Höhe zu addieren, um die beobachtete Höhe zu erhalten. Die Refraktion hängt auch vom Luftdruck und der Temperatur der Atmosphäre ab. Die angegebenen Werte sind deshalb als Mittelwerte zu verstehen. Ausführliche Tabellen der Normalrefraktion oder geeignete numerische Näherungsformeln sind in der entsprechenden Literatur zu finden.

Tabelle 3.2. Werte der Refraktion in Horizontnähe

h	10°	5°	2°	1°	0°
R	$5'31''$	$10'15''$	$19'7''$	$25'36''$	$34'$

Wir haben gesehen, daß wir in der Auf- und Untergangsrechnung von topozentrischen Höhen ausgehen müssen. Zusätzlich müssen wir die Refraktion am Horizont in Rechnung stellen. Weil die Auf- und Untergangszeiten immer auf den oberen Rand von Sonne oder Mond bezogen werden, haben wir noch den scheinbaren Radius s_\odot oder s_{Mond} zu berücksichtigen. Zum Zeitpunkt des Auf- oder Untergangs hat das Gestirn also die *geozentrische* Höhe

$$h_{A/U} = 0^\circ + \pi - R_{h=0} - s . \quad (3.10)$$

Unsere Aufgabe besteht nun darin, für ein vorgegebenes Datum den Zeitpunkt zu finden, zu dem der betrachtete Himmelskörper die Höhe $h = h_{A/U}$ erreicht. Weil es keinen Sinn hat, die solchermaßen erhaltenen Zeiten genauer als auf wenige Minuten angeben zu wollen, erspart man sich die genaue Berechnung von π und s und verwendet stattdessen Mittelwerte. Üblicherweise legt man deshalb den Berechnungen der Auf- und Untergangszeiten folgende Werte zugrunde:

- Sonnenauf- oder -untergang : $h_{A/U} = -0^\circ 50'$,
- Mondauf- oder -untergang : $h_{A/U} = +0^\circ 08'$ und
- bei Sternen oder Planeten : $h_{A/U} = -0^\circ 34'$.

Da unser Programm auch die Zeiten des Dämmerungsanfangs und -endes berechnet, soll hier noch auf die gebräuchlichen Definitionen eingegangen werden. Man unterscheidet

- die *astronomische* Dämmerung bei $h_\odot = -18^\circ$,
- die *nautische* Dämmerung bei $h_\odot = -12^\circ$ und
- die *bürgerliche* Dämmerung bei $h_\odot = -6^\circ$.

An diesen Werten werden keine Korrekturen für Parallaxe, Refraktion oder scheinbaren Halbmesser mehr angebracht. Die Dämmerung beginnt oder endet also definitionsgemäß, wenn die *geozentrische* Sonnenhöhe einen der genannten Werte annimmt.

Es bleibt nun das allgemeine Problem zu lösen, wie man den Zeitpunkt findet, zu dem ein Gestirn eine vorgegebene Höhe erreicht.

3.6 Auf- und Untergänge

Gleichung (3.5) gestattet uns, bei Kenntnis der geographischen Breite φ , der Deklination δ und des Stundenwinkels τ die Höhe h eines Sterns zu ermitteln. Den Stundenwinkel erhalten wir dabei zu bekannter Ortssternzeit aus (3.8).

Geben wir dagegen die Höhe h vor, dann können wir (3.5) umformen und erhalten daraus eine Beziehung für den Stundenwinkel:

$$\cos \tau = \frac{\sin h - \sin \varphi \sin \delta}{\cos \varphi \cos \delta} . \quad (3.11)$$

Man kann damit den Stundenwinkel berechnen, den ein Stern hat, wenn er in der Höhe h über dem Horizont beobachtet wird. Die Gleichung ist aber nicht für alle beliebigen Werte von h definiert, sondern nur für solche Höhen, die der Stern auch tatsächlich erreichen kann. Bei einer geographischen Breite von $\varphi = 50^\circ$ ist ein Stern der Deklination $\delta = 60^\circ$ zum Beispiel Zirkumpolarstern und nur zwischen $h = 20^\circ$ und $h = 80^\circ$ zu beobachten. Ist der Wert für $\cos \tau$ bei einer vorgegebenen Höhe h betragsmäßig größer als 1, so heißt dies, daß der Stern immer oberhalb oder immer unterhalb dieser Höhe zu finden ist.

Für Sternauf- und -untergänge setzt man die Höhe $h_{A/U} = -0^\circ 34'$ in (3.11) ein und drückt den erhaltenen Stundenwinkel im Zeitmaß ($15^\circ \cong 1^h$) aus. Er gibt an, wieviele Stunden Sternzeit ein eben aufgehender Stern bis zu seiner Kulmination benötigt. Da ein Sterntag eine Länge von $23^h 56^m 4^s 091$ Sonnenzeit hat, ergibt sich ein Faktor von $23^h 56^m 4^s 091 / 24^h = 0.9972696$, mit dem ein in Sternzeit gegebenes Zeitintervall in Sonnenzeit umgerechnet werden kann. Multipliziert man den Stundenwinkel mit diesem Faktor, dann erhält man den sogenannten *halben Tagbogen*. Darunter versteht man die Hälfte der Zeit, die der Stern insgesamt sichtbar ist.

Bei bekannter Rektaszension α_* des Sterns folgt aus der Beziehung

$$\Theta_{A/U} = \begin{cases} \alpha_* - \tau & \text{für den Aufgang} \\ \alpha_* + \tau & \text{für den Untergang} \end{cases}$$

die Sternzeit zum Zeitpunkt des Aufgangs bzw. des Untergangs. Bezeichnet man die Ortssternzeit um 0^h mit Θ_0 , dann geht der Stern

$$0.9973 \cdot (\Theta_0 - \Theta_A)$$

Stunden vor Mitternacht auf und

$$0.9973 \cdot (\Theta_U - \Theta_0)$$

Stunden nach Mitternacht unter.

Sonne und Mond verändern im Gegensatz zu den Sternen ihre Koordinaten im Laufe eines Tages merklich. Zur Berechnung des Stundenwinkels und der Auf- und Untergangszeiten nach den eben angegebenen Gleichungen benötigt man aber die Rektaszension und die Deklination im Moment des Horizontdurchgangs. Dies macht ein iteratives Vorgehen erforderlich. Man berechnet die Auf- und Untergangszeiten zunächst mit den Koordinaten zu einem beliebigen Zeitpunkt des jeweiligen Tages (meist wählt man $t = 12^h$). Mit den so ermittelten Näherungen

lassen sich verbesserte Koordinaten und daraus wieder genauere Auf- und Untergangszeiten bestimmen. Im Fall des Mondes werden diese Schritte wiederholt, bis sich die erhaltenen Zeiten um weniger als etwa eine Minute unterscheiden. Bei der Sonne und den Planeten liefert bereits die erste Iteration ein ausreichend genaues Ergebnis.

Leider gibt es besonders bei der Bestimmung der Mondauf- und -untergangszeiten einige Problemfälle, auf die sich das einfache Iterationsverfahren schlecht anwenden läßt. Hier wären folgende Punkte zu nennen:

- Da der Mond bei seinem Umlauf um die Erde von West nach Ost weiterwandert, steht er länger als Sterne mit derselben Deklination am Himmel. Dadurch verspäten sich die Mondaufgänge im Durchschnitt um etwa 50^m pro Tag. Dies hat zur Folge, daß es in jedem Monat einen Tag gibt, an dem kein Mondaufgang stattfindet und einen weiteren Tag, an dem der Mond nicht untergeht. Dazu ein Beispiel: der Mond geht am 8. Februar 1988 in München um 23^h30^m auf und tagsdarauf um 9^h41^m wieder unter. Durch seine Bewegung relativ zu den Sternen bleibt er dann bis nach Mitternacht unterhalb des Horizonts. Der nächste Mondaufgang findet damit nicht am 9. Februar, sondern erst am 10. Februar um 0^h43^m statt.
- In hohen geographischen Breiten kommt es vor, daß Sonne und Mond oft tagelang über oder unter dem Horizont stehen. Die Auf- und Untergangszeiten werden dann wesentlich durch die tägliche Änderung der Deklination bestimmt. Mögliche streifende Horizontberührungen sind dabei mit dem Iterationsverfahren nur schwer zu erfassen.

Aus diesen Gründen basiert das Programm **Sunset** auf einem anderen Prinzip. Die Auf- und Untergangszeiten werden hierbei durch inverse Interpolation einer Folge von Sonnen- und Mondhöhen ermittelt. Dies erfordert zwar einen höheren Rechenaufwand, führt aber zu einer wesentlichen Vereinfachung der Programmstruktur.

3.7 Quadratische Interpolation

Berechnet man eine Tabelle der Höhen in stündlichem Abstand, dann ist es möglich, den Verlauf der Höhe als Funktion der Zeit durch eine einfache interpolierende Funktion darzustellen. Für unsere Zwecke ist die quadratische Interpolation geeignet: aus drei Funktionswerten lassen sich die Koeffizienten einer Parabel berechnen, die den Funktionsverlauf zwischen den gegebenen Stützpunkten annähert. Dazu geht man von drei Funktionswerten

$$y_- = f(x = -1), \quad y_0 = f(x = 0) \quad \text{und} \quad y_+ = f(x = +1)$$

aus. Nun sucht man die Koeffizienten a , b und c einer Parabel

$$y = a \cdot x^2 + b \cdot x + c \quad , \tag{3.12}$$

die durch die Punkte $(-1, y_-)$, $(0, y_0)$ und $(1, y_+)$ verläuft. Eingesetzt in die Parabelgleichung ergeben sich die drei Gleichungen

$$\begin{aligned}y_- &= a - b + c \\y_0 &= c \\y_+ &= a + b + c\end{aligned},$$

aus denen sich die gesuchten Parabelkoeffizienten bestimmen lassen:

$$\begin{aligned}a &= (y_+ + y_-)/2 - y_0 \\b &= (y_+ - y_-)/2 \\c &= y_0\end{aligned}. \tag{3.13}$$

Die so gefundene Parabel hat für $b^2 \geq 4ac$ die Nullstellen

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{3.14}$$

und das Extremum

$$\begin{aligned}x_E &= \frac{-b}{2a} \\y_E &= a \cdot x_E^2 + b \cdot x_E + c\end{aligned}. \tag{3.15}$$

Die Funktion Quad wertet diese Gleichungen aus und wählt zusätzlich genau die Nullstellen aus, die zwischen $x = -1$ und $x = +1$ liegen.

```
//-----
// Quad: Quadratische Interpolation
//      Bestimmt Nullstellen und Extrema anhand von drei Funktionswerten zu
//      aequidistanten Stuetzstellen
//      y_minus  Funktionswert an der Stelle x = -1
//      y_0       Funktionswert an der Stelle x = 0
//      y_plus   Funktionswert an der Stelle x = 1
//      xe        Abszisse des Extremums (kann ausserhalb von [-1, 1] liegen)
//      ye        Funktionswert bei xe
//      root1     Erste gefundene Nullstelle
//      root2     Zweite gefundene Nullstelle
//      n_root    Anzahl der im Intervall [-1, 1] gefundenen Nullstellen
//-----
void Quad ( double y_minus, double y_0, double y_plus,
            double& xe, double& ye, double& root1,
            double& root2, int& n_root )
{
    double a,b,c, dis, dx;
    n_root = 0;
    // Koeffizienten der interpolierenden Parabel y=a*x^2+b*x+c
    a = 0.5*(y_plus+y_minus) - y_0;
    b = 0.5*(y_plus-y_minus);
    c = y_0;
    // Bestimme das Extremum
    xe = -b/(2.0*a);
    ye = (a*xe+b) * xe + c;
    dis = b*b - 4.0*a*c; // Diskriminante von y=a*x^2+b*x+c
```

```

if (dis >= 0) // Die Parabel hat reelle Nullstellen
{
    dx = 0.5 * sqrt (dis) / fabs (a);
    root1 = xe - dx;
    root2 = xe + dx;
    if (fabs(root1) <= 1.0) ++n_root;
    if (fabs(root2) <= 1.0) ++n_root;
    if (root1 < -1.0) root1 = root2;
}
}

```

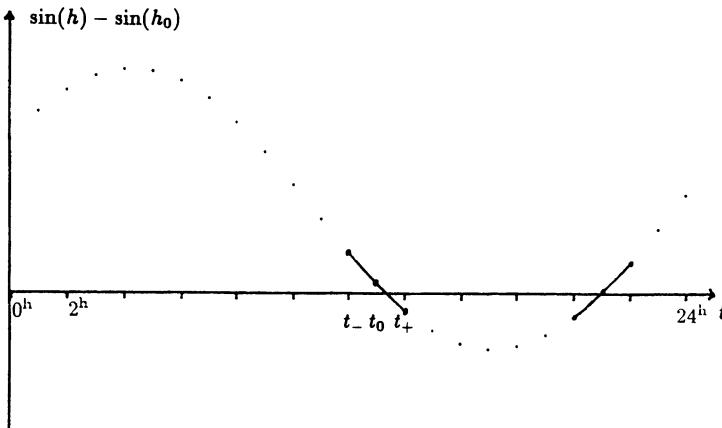


Abb. 3.4. Bestimmung der Auf- und Untergangszeiten

3.8 Das Programm SUNSET

Das Programm **Sunset** berechnet die Auf- und Untergangszeiten von Sonne und Mond sowie Beginn und Ende der Dämmerung in einem Zeitraum von 10 Tagen. Einzugeben sind dazu das Startdatum, die geographischen Koordinaten des Beobachtungsortes und die Differenz zwischen Zonen- und Weltzeit.

Die Suche nach den Zeiten der verschiedenen Ereignisse erfolgt nach dem in Abb. 3.4 dargestellten Schema. Mit Hilfe der Funktion **SinAlt** wird der Sinus der Sonnen- oder Mondhöhe in stündlichen Abständen berechnet. Diese Werte werden in Quad interpoliert und auf Nullstellen untersucht. Findet sich eine Nullstelle und standen Sonne oder Mond zu Beginn des Tages unter dem Horizont, so handelt es sich dabei um einen Aufgang. Andernfalls hat man den Zeitpunkt des Untergangs gefunden. Schließlich kann es noch vorkommen, daß zwei Nullstellen im Suchintervall gefunden werden. Um hier entscheiden zu können, welcher der beiden Zeitpunkte den Aufgang oder den Untergang darstellt, wird geprüft, ob das Extremum der Höhe unter oder über dem Horizont liegt. Die Suche wird gegebenenfalls fortgesetzt, bis das Ende des Tages erreicht ist. Dann kann auch entschieden werden, ob der betrachtete Himmelskörper zirkumpolar ist oder ganztägig unter dem Horizont steht.

Die Zeiten der bürgerlichen, nautischen oder astronomischen Dämmerung werden in gleicher Weise berechnet, indem vom Sinus der Horizonthöhe ein konstanter Wert $\sin(-6^\circ)$, $\sin(-12^\circ)$ oder $\sin(-18^\circ)$ subtrahiert wird.

```
-----
// Datei: Sunset.cpp
// Zweck: Auf- und Untergangszeiten von Sonne und Mond sowie Daemmerungszeiten
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
-----

#include <cmath>
#include <iomanip>
#include <iostream>
#include "APC_Const.h"
#include "APC_Math.h"
#include "APC_Moon.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"

using namespace std;

// zu suchende Ereignisse
enum enEvent {
    Moon,        // Mondauf- oder -untergang
    Sun,         // Sonnenauf- oder -untergang
    CivilTwi,    // buergerliche Daemmerung
    NautiTwi,    // nautische Daemmerung
    AstroTwi    // astronomische Daemmerung
};

//-----
// SinAlt: Sinus der Hoehe von Sonne oder Mond
// Event      zu suchendes Ereignis
// MJD0       Oh am Berechnungsdatum (als MJD)
// Hour        Stunde
// lambda     Geographische Laenge (oestl. pos.) in [rad]
// Cphi       Kosinus der geographischen Breite
// Sphi       Sinus der geographischen Breite
// <return>   Sinus der Hoehe von Sonne/Mond zum Zeitpunkt des Ereignisses
//-----
double SinAlt ( enEvent Event, double MJD0, double Hour,
                double lambda, double Cphi, double Sphi )
{
    double MJD, T, RA, Dec, tau;

    MJD = MJD0 + Hour/24.0;
    T   = (MJD-51544.5)/36525.0;
    if ( Event == Moon )
        MiniMoon ( T, RA, Dec );
    else
        MiniSun ( T, RA, Dec );
    tau = GMST(MJD) + lambda - RA;
    return ( Sphi*sin(Dec)+Cphi*cos(Dec)*cos(tau) );
}
```

```

//-----
// GetInput: Abfrage der Eingabedaten
//   MJD      Zeit als MJD
//   lambda   Geographische Laenge des Beobachters (oestl. pos.) in [rad]
//   phi      Geographische Breite des Beobachters in [rad]
//   zone     Differenz Zonenzeit - Weltzeit in [d]
//   twilight  bezeichnet buergerliche, nautische oder astronomische Daemmerung
//-----
void GetInput ( double& MJD, double& lambda,
                double& phi, double& zone, enEvent& twilight )
{
    int year, month, day;
    char cTwilight;
    cout << " Startdatum (JJJJ MM TT)           ... ";
    cin >> year >> month >> day; cin.ignore(81,'\'n');
    cout << endl;
    cout << " Beobachtungsort: Laenge (oestl. pos.) [Grad] ... ";
    cin >> lambda; cin.ignore(81,'\'n');
    cout << "                                Breite [Grad]       ... ";
    cin >> phi; cin.ignore(81,'\'n');
    cout << "                                Zonenzeit - UT [h]   ... ";
    cin >> zone; cin.ignore(81,'\'n');
    cout << " Daemmerungsdefinition (b, n oder a) ... ";
    cin >> cTwilight; cin.ignore(81,'\'n');
    switch (tolower(cTwilight)) {
        case 'b': twilight = CivilTwi; break;
        case 'a': twilight = AstroTwi; break;
        case 'n':
        default: twilight = NautiTwi;
    }
    lambda*=Rad; phi*=Rad; zone/=24.0;
    MJD = Mjd(year,month,day) - zone;
}

//-----
// FindEvents: Sucht Aufgangs-, Untergangs- oder Daemmerungereignisse
//   Event    zu suchendes Ereignis
//   MJDOh   Oh Zonenzeit am Berechnungsdatum als MJD
//   lambda   Geographische Laenge des Beobachters (oestl. pos.) in [rad]
//   phi     Geographische Breite des Beobachters in [rad]
//   LT_Rise  Zonenzeit fuer Aufgang oder Morgendaemmerung
//   LT_Set   Zonenzeit fuer Untergang oder Abenddaemmerung
//   rises    findet das Ereignis statt?
//   sets     findet das Ereignis statt?
//   above    Sonne oder Mond ist zirkumpolar
//-----
void FindEvents ( enEvent Event, double MJDOh, double lambda, double phi,
                  double& LT_Rise, double& LT_Set,
                  bool& rises, bool& sets, bool& above )
{
    static const double sinh0[5] = {
        sin(Rad*( +8.0/60.0)), // Mondauf/-untergang bei h=+8'
        sin(Rad*(-50.0/60.0)), // Sonnenauf/-untergang bei h=-50'
        sin(Rad*(- 6.0  )), // Buergerliche Daemmerung bei h=-6 Grad
}

```

```

sin(Rad*(-12.0)), // Nautische Daemmerung bei h=-12 Grad
sin(Rad*(-18.0)), // Astronomische Daemmerung bei h=-18 Grad
};

const double Cphi = cos(phi);
const double Sphi = sin(phi);
double hour = 1.0;
double y_minus, y_0, y_plus;
double xe, ye, root1, root2;
int nRoot;

// Initialisierung
y_minus = SinAlt(Event, MJD0h, hour-1.0, lambda, Cphi, Sphi) - sinh0[Event];
above = (y_minus>0.0); rises = false; sets = false;

// Schleife ueber Suchintervalle von [0h-2h] bis [22h-24h]
do {
    y_0 = SinAlt( Event, MJD0h, hour , lambda, Cphi, Sphi )-sinh0[Event];
    y_plus = SinAlt( Event, MJD0h, hour+1.0, lambda, Cphi, Sphi )-sinh0[Event];
    // Bestimme Parabel durch die drei Werte y_minus,y_0,y_plus
    Quad ( y_minus, y_0, y_plus, xe, ye, root1, root2, nRoot );
    if ( nRoot==1 ) {
        if ( y_minus < 0.0 ) { LT_Rise = hour+root1; rises = true; }
        else { LT_Set = hour+root1; sets = true; }
    }
    if ( nRoot == 2 ) {
        if ( ye < 0.0 ) { LT_Rise = hour+root2; LT_Set = hour+root1; }
        else { LT_Rise = hour+root1; LT_Set = hour+root2; }
        rises = true; sets = true;
    }
    y_minus = y_plus; // Vorbereitung fuer das naechste Intervall
    hour += 2.0;
}
while ( !( ( hour == 25.0 ) || ( rises && sets ) ) );
}

//-----
// 
// Hauptprogramm
// 
//-----
void main()
{
    bool above, rise, sett;
    int iEvent, day;
    double lambda, zone, phi;
    double date, start_date, LT_Rise, LT_Set;
    enEvent Event, Twilight;

    // Titel
    cout << endl
        << " SUNSET: Auf- und Untergangszeiten von Sonne und Mond" << endl
        << " (c) 1999 Oliver Montenbruck, Thomas Pfleger " << endl
        << endl;
    // Abfrage der Eingabedaten
    GetInput (start_date, lambda, phi, zone, Twilight );
}

```

```

// Vorspann
cout << endl
    << "      Datum"
    << "          Mond           Sonne           Daemmerung " << endl
    << "          "
    << "          Auf-/Untergang   Auf-/Untergang   Anfang/Ende"
    << endl << endl;

// Schleife ueber 10 aufeinanderfolgende Tage
for (day=0; day<10; day++) {
    // Aktuelles Datum
    date = start_date + day;
    cout << " " << DateTime(date+zone) << " ";
    // Schleife ueber zu suchende Ereignisse
    for (iEvent=0; iEvent<=2; iEvent++) {
        // Nach Ereignissen fuer Sonne/Mond: Auswahl der Daemmerungsdefinition
        Event = (iEvent<2) ? (enEvent) iEvent : Twilight;
        // Versuche die Zeitpunkte der Ereignisse zu bestimmen
        FindEvents ( Event, date, lambda, phi,
                     LT_Rise, LT_Set, rise, sett, above );
        // Ausgabe
        if ( rise || sett ) {
            if ( rise )
                cout << "     " << Time(LT_Rise,HHMM) << " ";
            else
                cout << "     ----- ";
            if ( sett )
                cout << "     " << Time(LT_Set,HHMM) << " ";
            else
                cout << "     ----- ";
        }
        else
            if ( above ) {
                if ( Event >= CivilTwi )
                    cout << "     immer hell   ";
                else
                    cout << "     immer sichtbar ";
            }
            else {
                if ( Event >= CivilTwi )
                    cout << "     immer dunkel  ";
                else
                    cout << "     immer unsichtbar";
            }
        cout << endl;
    }
}

// Nachspann
cout << endl;
cout << " alle Zeiten in Zonenzeit (=UT"
    << showpos << 24.0*zone << noshowpos << "h)"
    << endl;
}

```

Wir wollen im folgenden die Bedienung von *Sunset* an einigen Beispielen veranschaulichen und dabei Besonderheiten kennenlernen, die bei der Berechnung von Auf- und Untergangszeiten auftreten.

Sunset berechnet die gewünschten Zeiten für ein Intervall von jeweils zehn Tagen, das mit dem einzugebenden Datum beginnt. Die geographische Länge des Beobachtungsortes wird für östliche Längen, wie sie in Mitteleuropa vorkommen, positiv gezählt. Ergänzend ist noch die Differenz zwischen Zonenzeit und Weltzeit in Stunden anzugeben. Ein Wert von 1 bedeutet beispielsweise Mitteleuropäische Zeit (MEZ), ein solcher von 2 die Mitteleuropäische Sommerzeit (MESZ). Wünscht man die Ausgabedaten bezogen auf die Weltzeit, so gibt man hier eine 0 ein. Für die Zeitzonen Nordamerikas wäre ein negativer Wert einzugeben. Schließlich kann über die Angabe der Dämmerungsdefinition ausgewählt werden, ob die Dämmerungszeiten für die bürgerliche („b“), nautische („n“) oder astronomische Dämmerung („a“) berechnet werden sollen.

Zunächst sollen die Auf- und Untergangszeiten der Sonne und des Mondes für München ($\lambda = 11^\circ 6$ Ost, $\varphi = 48^\circ 1$ Nord) ab dem 23. März 2000 berechnet werden. Wir wünschen die Ausgabe in Mitteleuropäischer Zeit (1^h Differenz zur Weltzeit). Zuerst muß das Startdatum in der Reihenfolge Jahr, Monat und Tag eingegeben werden, anschliessend folgen die geographischen Koordinaten und die Zonenzeitdifferenz. Bei der Frage nach der gewünschten Dämmerungsdefinition wählen wir die *nautische* Dämmerung, die bei einer Sonnenhöhe von -12° einsetzt. Alle Eingaben sind im folgenden Dialog durch kursive Schrift dargestellt. *Sunset* bestimmt aus unseren Angaben die Auf- und Untergangszeiten von Sonne und Mond sowie Anfang und Ende der Dämmerung.

SUNSET: Auf- und Untergangszeiten von Sonne und Mond
(c) 1999 Oliver Montenbruck, Thomas Pfleger

Startdatum (JJJJ MM TT) ... 2000 03 23

Beobachtungsort: Laenge (oestl. pos.) [Grad] ... +11.6
Breite [Grad] ... +48.1
Zonenzeit - UT [h] ... +1
Daemmerungsdefinition (b, n oder a) ... n

Datum	Mond		Sonne		Daemmerung	
	Auf-/Untergang	Auf-/Untergang	Auf-/Untergang	Auf-/Untergang	Anfang/Ende	Anfang/Ende
2000/03/23	22:12	08:01	06:10	18:31	05:02	19:39
2000/03/24	23:17	08:28	06:08	18:32	05:00	19:41
2000/03/25	----	08:58	06:06	18:34	04:58	19:42
2000/03/26	00:18	09:33	06:04	18:35	04:56	19:44
2000/03/27	01:16	10:13	06:02	18:37	04:53	19:46
2000/03/28	02:08	10:59	06:00	18:38	04:51	19:47
2000/03/29	02:55	11:51	05:58	18:40	04:49	19:49
2000/03/30	03:37	12:48	05:56	18:41	04:47	19:50
2000/03/31	04:13	13:51	05:54	18:43	04:45	19:52
2000/04/01	04:44	14:56	05:52	18:44	04:42	19:54

alle Zeiten in Zonenzeit (=UT+1h)

In der ausgegebenen Tabelle fällt auf, daß in der Spalte mit den Aufgangszeiten des Mondes am 25.3. keine Zeit angeführt ist. Der Mond geht in dieser Nacht erst nach Mitternacht auf, und damit fällt dieser Aufgang schon auf den 26.3. In der Regel gibt es in jedem Monat einen Tag, an dem der Mond nicht auf- oder untergeht.

Um weitere interessante Effekte kennenzulernen, berechnen wir die Auf- und Untergangszeiten für einen auf dem 65. Breitengrad gelegenen fiktiven Ort in Europa. Wir wählen als Startdatum den 15. Juni 1989 und die Mitteleuropäische Sommerzeit als Zonenzeit. Wir erhalten dann von Sunset folgende Ausgabe:

SUNSET: Auf- und Untergangszeiten von Sonne und Mond
(c) 1999 Oliver Montenbruck, Thomas Pfleger

Startdatum (JJJJ MM TT) ... 1989 06 15

Beobachtungsort: Laenge (oestl. pos.) [Grad] ... +10.0

Breite [Grad] ... +65.0

Zonenzeit - UT [h] ... +2.0

Daemmerungsdefinition (b, n oder a) ... n

Datum	Mond		Sonne		Daemmerung Anfang/Ende
	Auf-/Untergang		Auf-/Untergang		
1989/06/15	19:58	01:00	02:24	00:16	immer hell
1989/06/16	22:26	23:53	02:23	00:18	immer hell
1989/06/17	immer unsichtbar		02:22	00:19	immer hell
1989/06/18	immer unsichtbar		02:21	00:20	immer hell
1989/06/19	immer unsichtbar		02:20	00:21	immer hell
1989/06/20	immer unsichtbar		02:20	00:22	immer hell
1989/06/21	02:39	03:24	02:20	00:23	immer hell
1989/06/22	01:35	06:21	02:20	00:23	immer hell
1989/06/23	01:15	08:29	02:21	00:23	immer hell
1989/06/24	01:01	10:25	02:22	00:22	immer hell

alle Zeiten in Zonenzeit (=UT+2h)

Hier fällt auf, daß der Mond nach seinem Untergang am 16. Juni um 23^h53^m bis zum 21. Juni um 2^h39^m unter dem Horizont bleibt und damit unsichtbar ist. Ganz entsprechend ist es in höheren Breiten auch möglich, daß der Mond tagelang über dem Horizont steht.

Natürlich können ähnliche Verhältnisse auch bei der Sonne vorkommen. Man spricht dann von der Polarnacht oder vom Polartag. Mit Sunset kann man auch bestimmen, wie lange die Polarnacht an einem bestimmten Ort dauert. Dazu variiert man den Berechnungszeitraum solange, bis für die Sonne das erste Mal „immer unsichtbar“ ausgegeben wird. Am betreffenden Tag beginnt die Polarnacht. Das Ende der Polarnacht findet man in völlig analoger Weise.

Die Angabe „immer hell“ in der Spalte „Dämmerung“ bedeutet, daß es nicht mehr vollständig finster wird. Dies ist bereits in Breiten von 50° möglich, wenn man von der astronomischen Dämmerung ausgeht.

3.9 Das Programm PLANRISE

In den vorangegangenen Abschnitten haben wir das Rüstzeug für die Berechnung von Auf- und Untergangszeiten nach der Methode der quadratischen Interpolation kennengelernt. Damit lassen sich auch Sonderfälle der Auf- und Untergangsrechnung elegant behandeln, die ansonsten erhebliche Schwierigkeiten bereiten können. Wir wollen nun aber auch ein Programm vorstellen, das am Beispiel der Planeten die Auf- und Untergangszeiten nach der iterativen Methode berechnet. Der kommentierte Quelltext von **Planrise** ist auf der CD zum Buch zu finden.

Die Arbeitsweise von **Planrise** baut auf den in Abschn. 3.6 vorgestellten Gleichungen auf. Zur Berechnung der Positionen der großen Planeten wird auf die in Kap. 6 vorgestellte Funktion **PertPosition** zurückgegriffen. Sie basiert auf analytischen Reihenentwicklungen und liefert speziell für die äußeren Planeten wesentlich genauere Ergebnisse als die Annahme einfacher Keplerbahnen.

Zur Berechnung der Auf- und Untergangszeiten eines Planeten ermitteln wir zunächst für eine beliebige Zeit am fraglichen Tag seine (geozentrischen) äquatorialen Koordinaten α und δ . Aus (3.11) folgt dann der Wert $\cos \tau(\varphi, \delta)$ für den Stundenwinkel zum Zeitpunkt des Auf- oder Untergangs. Erhält man bei der Auswertung der rechten Seite dieser Gleichung einen Wert, dessen Betrag größer als eins ist, so geht der Planet den ganzen Tag über nicht auf oder unter. Zur Entscheidung, welcher Fall vorliegt, vergleicht **Planrise** die Deklination des Planeten mit der geographischen Breite des Beobachters.

Da sich bei den Planeten Horizontalparallaxe und scheinbarer Durchmesser praktisch nicht auf die Auf- und Untergangszeiten auswirken, setzt man für die geometrische Höhe beim Auf- und Untergang den Wert $h_{A/U} = -0^\circ 34'$ ein, der nur die Refraktion berücksichtigt. Die Iterationsgleichung für die Auf- bzw. Untergangszeiten lautet dann

$$t_{i+1} = t_i - 0.9973(\Theta(t_i) - \alpha(t_i) \pm \tau(\varphi, \delta(t_i))) \quad (3.16)$$

mit

$$\tau = \arccos \frac{\sin h_{A/U} - \sin \varphi \sin \delta}{\cos \varphi \cos \delta}. \quad (3.17)$$

Für die Berechnung der Aufgangszeit ist dabei das positive Vorzeichen zu verwenden. $\Theta(t_i)$ und $\alpha(t_i)$ bezeichnen die lokale Sternzeit und die Rektaszension des Planeten zum Zeitpunkt t_i der i -ten Näherung. Der Umrechnungsfaktor 0.9973 dient wie bereits erläutert zur Umrechnung zwischen Weltzeit und Sternzeit.

Analog zur Berechnung der Auf- und Untergangszeiten kann darüberhinaus aus der Beziehung

$$t_{i+1} = t_i - 0.9973(\Theta(t_i) - \alpha(t_i)) \quad (3.18)$$

die Kulminationszeit eines Planeten bestimmt werden.

Um die Rechenzeiten von **Planrise** zu verkürzen, werden einige Möglichkeiten zur Vereinfachung genutzt. So werden die Planetenkoordinaten α und δ sowie die Sternzeit Θ linear interpoliert, wobei als Stützstellen nur die jeweiligen Werte um 0^h und um 24^h des fraglichen Tages berechnet werden.

Als Anwendungsbeispiel sollen die Auf- und Untergangszeiten der Planeten am 31. Dezember 1999 in München ($\lambda = 11.6^\circ$ Ost, $\varphi = 48.1^\circ$ Nord) berechnet werden. Die Eingaben für Planrise sind kursiv gesetzt.

PLANRISE: Auf- und Untergangszeiten der Planeten und der Sonne
(c) 1999 Oliver Montenbruck, Thomas Pfleger

Berechnungsdatum (JJJJ MM TT) ... 1999 12 31

Beobachtungsort: Laenge (oestl. pos.) [Grad] ... +11.6
Breite [Grad] ... +48.1
Zonenzeit - UT [h] ... +1.0

Aufgang Kulmination Untergang

	Aufgang	Kulmination	Untergang
Sonne	08:04	12:16	16:29
Merkur	07:33	11:37	15:41
Venus	04:52	09:30	14:08
Mars	10:33	15:35	20:37
Jupiter	12:29	19:10	01:55
Saturn	13:09	20:10	03:14
Uranus	10:02	14:45	19:28
Neptun	09:25	13:57	18:29
Pluto	05:11	10:22	15:32

alle Zeiten in Zonenzeit (=UT+1h)

Gutes Wetter vorausgesetzt, sind in dieser Nacht also gleich drei helle Planeten (Mars, Jupiter und Saturn) mit bloßem Auge am Abendhimmel zu sehen.

4. Kometenbahnen

Während die Bahnen der großen Planeten oft schon über Jahre im voraus berechnet und veröffentlicht werden, beschränkt sich die Vorhersage von Kometenbahnen meist auf kurzfristige Ephemeriden und die Angabe der Bahnelemente. Die fünf bis sechs Bahnelemente beschreiben den räumlichen und zeitlichen Verlauf der Kometenbahn in wesentlich kürzerer Form, als eine lange Tabelle von Positionen. Bahnelemente neu entdeckter Kometen werden in der Regel zuerst von der Internationalen Astronomischen Vereinigung (IAU) in den IAU-Zirkularen publiziert. Daneben gibt es eine Reihe von Katalogen, in denen entsprechende Daten für periodische Kometen und Kleinplaneten verzeichnet sind.

Mit Hilfe der Bahnelemente und eines entsprechenden Programms kann man sich den Ort des Kometen relativ zu Erde und Sonne jederzeit selbst berechnen. Das Programm *Comet*, das in diesem Kapitel vorgestellt wird, kann nicht nur elliptische Bahnen wie die der Planeten und Asteroiden berechnen. Es eignet sich darüber hinaus für die bei Kometen auftretenden parabolischen, annähernd parabolischen und hyperbolischen Bahnen.

Die folgenden Abschnitte behandeln die Bewegung des Kometen relativ zur Sonne und zur Ekliptik, wobei Störungen durch die großen Planeten allerdings vernachlässigt werden. Ihre Berücksichtigung würde einen erheblichen Mehraufwand bedeuten. Die verschiedenen Koordinatentransformationen, die zur Umrechnung in geozentrische, äquatoriale Koordinaten benötigt werden, wurden bereits in Kap. 2 beschrieben und können dort bei Bedarf noch einmal nachgeschlagen werden.

4.1 Form und Lage der Bahn

Die Bewegung der Kometen um die Sonne erfüllt die gleichen Gesetze wie die der großen Planeten:

1. Die Form der Bahn ist eine Ellipse, Parabel oder Hyperbel (also ein Kegelschnitt), in deren einem Brennpunkt die Sonne steht. Dies bedeutet insbesondere, daß die Bahn in einer festen Ebene verläuft.
2. In gleichen Zeiten überstreicht die Verbindungsline Sonne-Komet gleiche Flächen (sogenannter *Flächensatz*).

Diese Gesetze wurden von Johannes Kepler gefunden und konnten später auf das allgemeine Gravitationsgesetz von Newton zurückgeführt werden. Allerdings gelten die Keplerschen Gesetze nur für den Fall, daß der Einfluß der großen Plane-

ten auf den Kometen gegen die Anziehungskraft der Sonne vernachlässigt werden kann.

Die Tatsache, daß die Bahn in einer Ebene verläuft, hängt damit zusammen, daß die Gravitationskraft der Sonne immer entlang der Richtung zum Kometen wirkt (sogenannte Zentralkraft). In einem kleinen Zeitintervall Δt bewegt sich der Komet mit der Geschwindigkeit $\dot{\mathbf{r}}$ um die Strecke $\dot{\mathbf{r}} \cdot \Delta t$ vom Ort \mathbf{r} zum Ort $\mathbf{r} + \dot{\mathbf{r}} \cdot \Delta t$ (Abb. 4.1). Zusammen mit der Sonne legen diese beiden Orte im Raum eine Ebene fest. Da die Kraft der Sonne aber immer nur in dieser Ebene wirkt, bleibt die Bewegung des Kometen auch weiterhin auf die anfängliche Bahnebene beschränkt. Auch der Flächensatz ist eine unmittelbare Folge der Zentralkraft. Der Beweis dafür ist allerdings mit ein wenig Mathematik verbunden und soll hier nicht gezeigt werden.

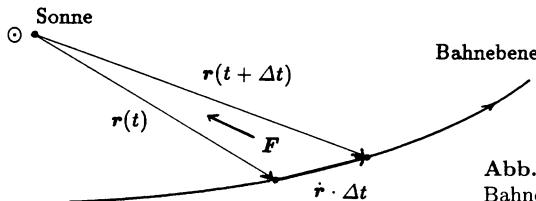


Abb. 4.1. Die Kraft auf den Kometen läßt die Bahnebene unverändert

Die genaue Form der Bahn wird durch die quadratische Abnahme der Gravitationskraft mit der Entfernung ($F \sim 1/r^2$) bewirkt. Die drei Grundformen – Ellipse, Parabel und Hyperbel – sind in Abb. 4.2 gegenübergestellt. Während elliptische Bahnen immer wieder periodisch durchlaufen werden, kommen Kometen auf parabolischen und hyperbolischen Bahnen nur einmal für kurze Zeit in die Nähe der Sonne. Der Bahnpunkt mit dem geringsten Abstand zur Sonne wird als *Perihel* bezeichnet. Die Verbindungsline Sonne-Perihel (sog. *Apsidenlinie*) teilt die Bahn in zwei symmetrische Hälften.

Der Ort in der Bahn wird durch die Entfernung r von der Sonne und die wahre Anomalie¹ ν gekennzeichnet. Darunter versteht man den Winkel zwischen der Richtung Sonne-Komet und der Richtung Sonne-Perihel. Den Zusammenhang zwischen r und ν gibt die Kegelschnittsgleichung wieder:

$$r = \frac{p}{1 + e \cdot \cos(\nu)} . \quad (4.1)$$

Die Größe p , das ist die Entfernung für $\nu = 90^\circ$, heißt *Bahnparameter*. Sie definiert die Ausdehnung der Bahn. e ist die sogenannte *Exzentrizität*, die angibt, wie stark die Bahn von der Kreisform ($e = 0$) abweicht. Höhere Werte bezeichnen immer langgestrecktere Ellipsen, die schließlich in Parabeln ($e = 1$) und Hyperbeln ($e > 1$) übergehen.

Eine etwas gebräuchlichere Größe als der Bahnparameter ist die große Halbachse, die allerdings nur für $e \neq 1$ definiert ist:

$$a = \frac{p}{1 - e^2} , \quad p = a(1 - e^2) . \quad (4.2)$$

¹Unter Anomalie versteht man im Sprachgebrauch der Himmelsmechanik einen Winkel. Neben der wahren Anomalie kennt man noch die mittlere und die exzentrische Anomalie.

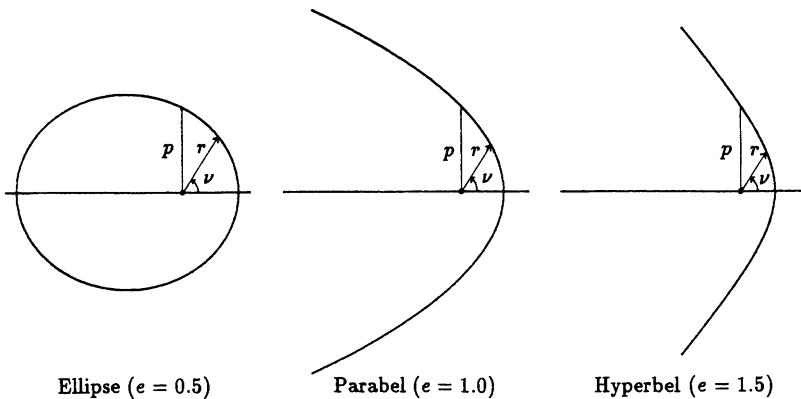


Abb. 4.2. Kegelschnitte der Exzentrizitäten $e = 0.5$, $e = 1.0$ und $e = 1.5$ mit gleichem Bahnpараметer p

Bei einer elliptischen Bahn hat a anschaulich die Bedeutung des Mittelwertes aus größter und kleinster Sonnenentfernung:

$$r_{\min} + r_{\max} = \frac{p}{1+e} + \frac{p}{1-e} = \frac{2p}{1-e^2} = 2a .$$

Zu beachten ist noch, daß die große Halbachse bei der Hyperbelbahn wegen $e > 1$ definitionsgemäß negativ ist. Für parabelnahe Bahnen ($e \approx 1$) verwendet man statt a die Periheldistanz $q = r_{\min}$:

$$q = \frac{p}{1+e} . \quad (4.3)$$

In einer exakt parabolischen Bahn ist $q = p/2$.

4.2 Der Ort in der Bahn

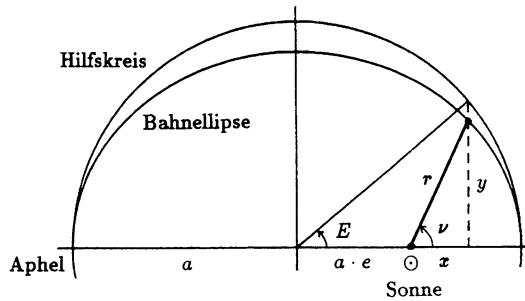
Die Kegelschnittsgleichung gibt zwar alle möglichen Orte an, an denen sich ein Komet während seines Umlaufs befinden kann, beantwortet aber nicht die Frage, zu welchen Zeiten er sich dort aufhält. Zur Lösung dieses Problems bedient man sich des Flächensatzes. Am Beispiel einer geschlossenen Ellipsenbahn sollen die wesentlichen Schritte dazu erläutert werden.

Eine Ellipse der Exzentrizität e erhält man, wenn man einen Kreis entlang einer Richtung um den Faktor $\sqrt{1-e^2}$ zusammendrückt. Die gesamte Fläche S einer Ellipse mit der großen Halbachse a ist daher um diesen Faktor kleiner als die eines Kreises mit Radius a :

$$S = (\pi a^2) \sqrt{1-e^2} .$$

Die Verbindungsstrecke Sonne-Komet überstreicht diese Fläche in der Umlaufszeit T und damit den Sektor

$$S(t) = (\pi a^2) \sqrt{1-e^2} \left(\frac{t-t_0}{T} \right) \quad (4.4)$$

Abb. 4.3. Zur Definition der exzentrischen Anomalie E

in der Zeit t nach dem Periheldurchgang t_0 . Wenn es nun gelingt, die Fläche dieses Sektors S durch den Bahnort (r, ν) darzustellen, dann hat man damit die gesuchte Beziehung zwischen dem Bahnort und der Zeit t gefunden. Im Fall einer Kreisbahn ist dies noch recht einfach, da S hier direkt proportional zum Winkel ν und zum Quadrat der Halbachse ist. Bei einer allgemeinen Ellipse muß man allerdings eine Hilfsgröße – die *exzentrische Anomalie* – einführen. Ihre geometrische Bedeutung ist aus Abb. 4.3 ersichtlich. Wählt man die x -Achse in der dort angegebenen Weise, dann hat ein Punkt P auf der Ellipse die x -Koordinate $x = a \cdot \cos E - a \cdot e$. Der Term $a \cdot e$ steht dabei für die Entfernung des Mittelpunktes der Ellipse vom Brennpunkt, in dem sich die Sonne befindet. Die y -Koordinate ist um den Stauchungsfaktor $\sqrt{1 - e^2}$ kleiner als die des Punktes P' auf dem Umkreis, also gleich $y = \sqrt{1 - e^2} \cdot a \cdot \sin E$. Damit gilt

$$\begin{aligned} x &= r \cos \nu = a (\cos E - e) \\ y &= r \sin \nu = a \sqrt{1 - e^2} \sin E . \end{aligned} \quad (4.5)$$

Zur Berechnung von S werden einige Hilfsflächen betrachtet, die in Abb. 4.4 noch einmal gesondert herausgezeichnet sind:

$$\begin{aligned} \text{Kreissektor } (P' O \Pi) : \quad S_1 &= \pi a^2 E / 360^\circ \\ \text{Ellipsensektor } (P O \Pi) : \quad S_2 &= \sqrt{1 - e^2} \cdot S_1 \\ \text{Dreieck } (O \odot P) : \quad S_3 &= \frac{1}{2} y \cdot (a \cdot e) = \frac{1}{2} a^2 e \sqrt{1 - e^2} \sin E \\ \text{Sektor } (P \odot \Pi) : \quad S &= S_2 - S_3 . \end{aligned}$$

Mit Hilfe der exzentrischen Anomalie läßt sich die Fläche S demnach in der Form

$$S(t) = \frac{1}{2} a^2 \sqrt{1 - e^2} \left(E \frac{\pi}{180^\circ} - e \sin E \right) \quad (4.6)$$

darstellen. Vergleicht man dies mit der obigen Formel für $S(t)$, dann erhält man als Beziehung zwischen der exzentrischen Anomalie und der Zeit die *Keplergleichung*:

$$E(t) - \frac{180^\circ}{\pi} e \sin E(t) = M(t) \quad (4.7)$$

mit

$$M(t) = 360^\circ \frac{(t - t_0)}{T} . \quad (4.8)$$

Um den Ort des Kometen in der Bahn zu einem beliebigen Zeitpunkt zu berechnen, muß man diese Gleichung nach E auflösen und den so erhaltenen Wert in die Gleichung (4.5) zur Bestimmung von x und y einsetzen. Im Abschnitt über die numerische Behandlung der Keplergleichung wird dies im einzelnen beschrieben.

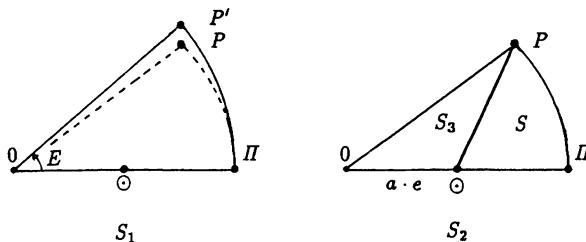


Abb. 4.4. Zur Herleitung der Sektorfläche S

Bisher wurde die Umlaufszeit als eine unabhängige Größe behandelt. Tatsächlich aber hängt sie über das *3. Keplersche Gesetz* mit der großen Halbachse zusammen:

$$\frac{a^3}{T^2} = \frac{GM_{\odot}}{4\pi^2} \quad \text{mit} \quad GM_{\odot} = 1.32712 \cdot 10^{20} \text{m}^3 \text{s}^{-2} . \quad (4.9)$$

Die Konstante auf der rechten Seite dieser Gleichung enthält das Produkt GM_{\odot} aus Gravitationskonstante und Sonnenmasse. Die Einheiten Meter (m) und Sekunde (s), in denen diese Größe hier angegeben ist, werden in der Praxis allerdings kaum verwendet. Die üblichen Einheiten sind ein Tag ($1 \text{ d} = 86400 \text{ s}$) und eine Astronomische Einheit ($1 \text{ AE} \approx 149.59787 \text{ Mio. km}$). Die AE bezeichnete ursprünglich den Wert der großen Halbachse der Erdbahn um die Sonne. Später entschloß sich die IAU dazu, die Astronomische Einheit so zu definieren, daß das Produkt aus Gravitationskonstante und Sonnenmasse den festgelegten Wert

$$\begin{aligned} GM_{\odot} &= k^2 \text{AE}^3 \text{d}^{-2} \quad \text{mit} \\ k &= 0.01720209895 \\ k^2 &\approx 2.959122083 \cdot 10^{-4} \end{aligned}$$

hat. k ist dabei die sogenannte Gaußsche Gravitationskonstante. Durch diese zunächst etwas befremdlich wirkende Formulierung ist die Definition der AE von eventuellen langfristigen Änderungen des Erdbahndurchmessers unabhängig.

Neben dem Ort des Kometen in der Bahn wird später auch seine Geschwindigkeit benötigt. Die Formeln dafür seien hier noch angegeben:

$$\begin{aligned} \dot{x} &= -\sqrt{\frac{GM_{\odot}}{a}} \frac{\sin E}{1 - e \cos E} \\ \dot{y} &= +\sqrt{\frac{GM_{\odot}(1 - e^2)}{a}} \frac{\cos E}{1 - e \cos E} . \end{aligned} \quad (4.10)$$

Für Hyperbelbahnen findet man ähnliche Formeln wie für Ellipsenbahnen. An die Stelle der Winkelfunktionen treten hier die Hyperbelfunktionen $\sinh x = (e^x - e^{-x})/2$ und $\cosh x = (e^x + e^{-x})/2$. Es gilt:

$$\begin{aligned} x = r \cos \nu &= |a| (e - \cosh H) \\ y = r \sin \nu &= |a| \sqrt{e^2 - 1} \sinh H \end{aligned} \quad (4.11)$$

und

$$\begin{aligned} \dot{x} &= -\sqrt{\frac{GM_{\odot}}{|a|}} \frac{\sinh H}{e \cosh H - 1} \\ \dot{y} &= +\sqrt{\frac{GM_{\odot}(e^2 - 1)}{|a|}} \frac{\cosh H}{e \cosh H - 1} . \end{aligned} \quad (4.12)$$

Die Größe H , die der exzentrischen Anomalie entspricht, genügt einer modifizierten Form der Keplergleichung:

$$e \sinh H - H = M_h = \sqrt{\frac{GM_{\odot}}{|a|^3}} \cdot (t - t_0) . \quad (4.13)$$

Bei Parabelbahnen ($e = 1$) drückt man zunächst x und y durch $\tan(\nu/2)$ aus:

$$\begin{aligned} x = r \cos \nu &= q \cdot \left(1 - \tan^2 \left(\frac{\nu}{2}\right)\right) \\ y = r \sin \nu &= 2q \cdot \tan \left(\frac{\nu}{2}\right) . \end{aligned} \quad (4.14)$$

q bezeichnet darin die früher eingeführte Periheldistanz. Die wahre Anomalie ν hängt über die *Barkersche Gleichung* von der Zeit ab:

$$\tan \left(\frac{\nu}{2}\right) + \frac{1}{3} \tan^3 \left(\frac{\nu}{2}\right) = \sqrt{\frac{GM_{\odot}}{2q^3}} (t - t_0) . \quad (4.15)$$

Es handelt sich dabei um eine Gleichung dritten Grades in $\tan(\nu/2)$. Im Gegensatz zur Keplergleichung kann sie zu gegebener Zeit t direkt nach ν aufgelöst werden. Setzt man

$$A = \frac{3}{2} \sqrt{\frac{GM_{\odot}}{2q^3}} (t - t_0) \quad \text{und} \quad B = \sqrt[3]{A + \sqrt{A^2 + 1}} ,$$

dann ist

$$\tan(\nu/2) = B - 1/B , \quad \nu = 2 \arctan(B - 1/B) .$$

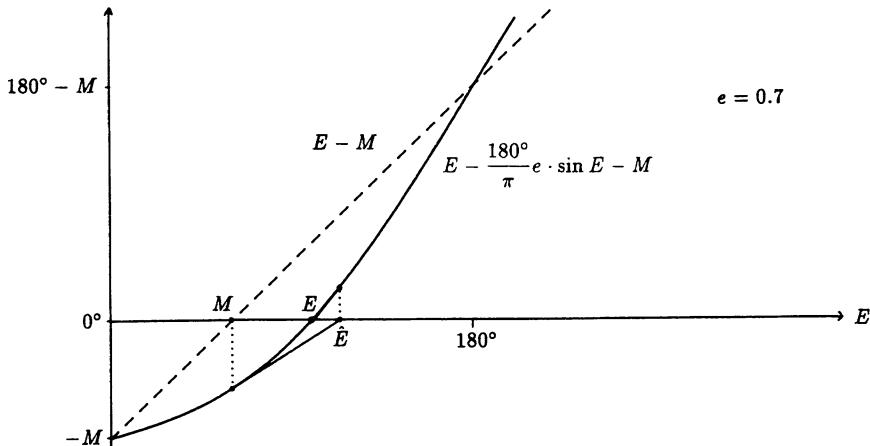


Abb. 4.5. Lösung der Keplergleichung mit Hilfe des Newtonverfahrens

4.3 Die numerische Behandlung der Keplergleichung

Die Keplergleichung lässt sich nicht analytisch nach der exzentrischen Anomalie als Funktion der mittleren Anomalie beziehungsweise der Zeit auflösen. Es gibt hierfür jedoch geeignete numerische Verfahren.

Zu einer gegebenen Zeit t berechnet man zunächst die mittlere Anomalie M . Für niedrige Exzentrizitäten unterscheidet sich diese nur wenig von der exzentrischen Anomalie E . Man kann sie deshalb als erste Näherung E_0 für eine anschließende Iteration verwenden. Formt man die Keplergleichung ein wenig um, dann erkennt man, daß zu ihrer Auflösung die Nullstelle der Funktion

$$f(E) = E - \frac{180^\circ}{\pi} e \sin E - M(t)$$

zu bestimmen ist. Eine bekannte Methode, die Nullstelle einer Funktion $f(x)$ zu finden, ist das *Newtonverfahren* (vgl. Abb. 4.5). Dazu nähert man die Funktion an einer Stelle x in der Umgebung der Nullstelle durch ihre Tangente mit der Steigung $f'(x)$ an und berechnet deren Schnittpunkt mit der x -Achse:

$$\hat{x} = x - \frac{f(x)}{f'(x)}$$

Im allgemeinen erhält man so eine verbesserte Näherung \hat{x} für die Nullstelle der Funktion f . Die Iterationsvorschrift hat im Fall der Keplergleichung die Gestalt

$$\hat{E} = E - \frac{E - (180^\circ/\pi) \cdot e \cdot \sin E - M(t)}{1 - e \cdot \cos E}$$

E ist dabei in Grad einzusetzen. Arbeitet man stattdessen im Bogenmaß, so ist der Faktor $180^\circ/\pi$ durch 1 zu ersetzen. Bei kreisnahen Bahnen erhält man in typischerweise rund drei Schritten eine auf etwa zehn Stellen genaue Lösung für E . Allerdings sollte man sich davor hüten, das Newtonverfahren unbesehen bei

hohen Exzentrizitäten anzuwenden. Es sei dem Leser an dieser Stelle nahegelegt, sich einmal selbst vom Verhalten der Iteration für $e = 0.99$ und $E_0 = M = 5^\circ$ zu überzeugen. Sicherheit bietet hier allerdings ein anderer Startwert, nämlich $E_0 = \pi = 180^\circ$. In der Funktion `EccAnom` wird dieser Startwert für Exzentrizitäten ab $e = 0.8$ eingesetzt. Die Iteration wird solange fortgesetzt, bis die Keplergleichung mit einer Genauigkeit erfüllt ist, die um zwei Dezimalen geringer als die relative Genauigkeit des `double` Zahlenformats ist. Die Maschinengenauigkeit kann dabei durch einen Aufruf der Funktion `numeric_limits<double>::epsilon` im Modul `<limits>` der C++ Standard Bibliothek zur Verfügung gestellt werden.

```
const double eps_mach = numeric_limits<double>::epsilon();

//-----
// EccAnom: Berechnung der exzentrischen Anomalie fuer elliptische Bahnen
//   M      Mittlere Anomalie in [rad]
//   e      Exzentrizitaet der Bahn [0,1[
// <return> Exzentrische Anomalie in [rad]
//-----

double EccAnom (double M, double e)
{
    const int    maxit = 15;
    const double eps    = 100.0*eps_mach;
    int        i=0;
    double E, f;
    // Startwert
    M = Modulo(M, 2.0*pi);  if (e<0.8) E=M; else E=pi;
    // Iteration
    do {
        f = E - e*sin(E) - M;
        E = E - f / ( 1.0 - e*cos(E) );
        ++i;
        if (i==maxit) { cerr << " Konvergenzprobleme in EccAnom" << endl; break; }
    }
    while (fabs(f) > eps);
    return E;
}
```

Mit Hilfe dieser Funktion kann die exzentrische Anomalie nun mit einem einzigen Aufruf bestimmt werden. Die restlichen Schritte, die zur Berechnung eines Bahnortes in der Ellipse benötigt werden, sind in der Funktion `Ellip` zusammengefaßt:

```
//-----
// Ellip: Berechnung der Orts- und Geschwindigkeitsvektoren fuer elliptische
//        Bahnen
//   GM      Produkt aus Gravitationskonstante und Zentralmasse [AE^3*d^-2]
//   M      Mittlere Anomalie in [rad]
//   a      Grosse Halbachse der Bahn in [AE]
//   e      Exzentrizitaet der Bahn (<1)
//   r      Ortsvektor bezogen auf die Bahnebene in [AE]
//   v      Geschwindigkeitsvektor bezogen auf die Bahnebene in [AE/d]
//-----

void Ellip ( double GM, double M, double a, double e,
             Vec3D& r, Vec3D& v )
```

```
{
    double k, E, cosE,sinE, fac, rho;
    k = sqrt(GM/a);
    E = EccAnom(M,e);
    cosE = cos(E);
    sinE = sin(E);
    fac = sqrt ( (1.0-e)*(1.0+e) );
    rho = 1.0 - e*cosE;
    r = Vec3D (a*(cosE-e), a*fac*sinE, 0.0);
    v = Vec3D (-k*sinE/rho, k*fac*cosE/rho, 0.0);
}
```

In entsprechender Weise lässt sich die Keplergleichung für Hyperbelbahnen lösen. Hier lautet die Iterationsvorschrift:

$$\hat{H} = H - \frac{e \cdot \sinh H - H - M_h(t)}{e \cdot \cosh H - 1}.$$

Ein geeigneter Startwert dazu ist

$$H_0 = \begin{cases} +\ln(+1.8 + 2M_h/e) & \text{für } M_h \geq 0 \\ -\ln(+1.8 - 2M_h/e) & \text{für } M_h < 0 \end{cases}$$

```
//------------------------------------------------------------------------------
// HypAnom: Berechnung der exzentrischen Anomalie fuer hyperbolische Bahnen
//      Mh      Mittlere Anomalie in [rad]
//      e       Exzentrizitaet der Bahn (>1)
//      <return> Exzentrische Anomalie in [rad]
//-----
double HypAnom (double Mh, double e)
{
    const int    maxit = 15;
    const double eps    = 100.0*eps_mach;
    int         i=0;
    double H, f;
    // Startwert
    H = log (2.0*fabs(Mh)/e + 1.8);
    if (Mh < 0.0) H = -H;
    // Iteration
    do {
        f = e*sinh(H) - H - Mh;
        H = H - f / ( e*cosh(H) - 1.0 );
        ++i;
        if (i==maxit) {
            cerr << " Konvergenzprobleme in HypAnom" << endl;
            break;
        }
    }
    while ( fabs(f) > eps*(1.0+fabs(H+Mh)) );
    return H;
}
```

Die rechtwinkeligen Koordinaten des Kometen in der Bahn kann man mit der Funktion `Hyperb` nach den Formeln des vorangehenden Abschnitts berechnen.

Anders als bei **Ellip** werden der Zeitpunkt des Periheldurchgangs und der Berechnungszeitpunkt anstelle der mittleren Anomalie M_h als Eingaben verlangt. Der Grund für diese unterschiedliche Wahl der Übergabegrößen liegt darin, daß die Funktion **Ellip** auch für die Berechnung von Asteroiden- und Planetenbahnen geeignet sein soll. Bei diesen Himmelskörpern wird aber im Gegensatz zu den Kometenbahnen nur selten die Perihelzeit als Bahnelement verwendet. Statt dessen findet man meist Angaben über die mittlere Anomalie M_{Ep} und deren tägliche Änderung n zu einer bestimmten Epoche t_{Ep} . Die mittlere Anomalie $M = M_{Ep} + n(t - t_{Ep})$ zu einer beliebigen Zeit t kann dann direkt als Eingabe für **Ellip** benutzt werden. Hyperbelbahnen treten dagegen überhaupt nur bei Kometenbahnen auf, bei denen wiederum die Angabe der Perihelzeit als Bahnelement üblich ist.

```
-----
// Hyperb: Berechnung der Orts- und Geschwindigkeitsvektoren fuer hyperbolische
//          Bahnen
// GM      Produkt aus Gravitationskonstante und Zentralmasse [AE^3*d^-2]
// t0      Zeitpunkt des Periheldurchgangs
// t      Berechnungszeitpunkt
// a      Grosse Halbachse der Bahn in [AE]
// e      Exzentrizitaet der Bahn (>1)
// r      Ortsvektor bezogen auf die Bahnebene in [AE]
// v      Geschwindigkeitsvektor bezogen auf die Bahnebene in [AE/d]
// Beachte: t0 und t in Julianischen Jahrhunderten seit J2000
// -----
void Hyperb ( double GM, double t0, double t, double a, double e,
              Vec3D& r, Vec3D& v )
{
    double k, Mh, H, coshH, sinhH, rho, fac;
    a = fabs(a);
    k = sqrt(GM/a);
    Mh = k*(t-t0)/a;
    H = HypAnom(Mh,e);
    coshH = cosh(H);
    sinhH = sinh(H);
    fac = sqrt( (e+1.0)*(e-1.0) );
    rho = e*coshH - 1.0;
    r = Vec3D (a*(e-coshH), a*fac*sinhH, 0.0);
    v = Vec3D (-k*sinhH/rho, k*fac*coshH/rho, 0.0);
}
```

4.4 Parabelnahe Bahnen

Im letzten Abschnitt wurde bereits angedeutet, daß die Berechnung von elliptischen oder hyperbolischen Kometenbahnen mit Exzentrizitäten in der Nähe von Eins bei kleinen Anomalien nicht immer ganz einfach ist. Man kann diese Schwierigkeiten aber von vornherein umgehen, wenn man sich zunutze macht, daß die Bahnen unter den genannten Voraussetzungen große Ähnlichkeit mit einer Parabel haben. Die hier gezeigte Methode geht auf den Himmelsmechaniker Karl Stumpff zurück.

Aus der Keplergleichung

$$E(t) - e \sin E(t) = \sqrt{\frac{GM_{\odot}}{a^3}} \cdot (t - t_0) ,$$

die hier im Bogenmaß ($180^\circ = \pi$) formuliert ist, folgt mit

$$a = q/(1 - e) \quad \text{und} \quad c_3 = (E - \sin E)/E^3$$

zunächst

$$(1 - e) \cdot E + e \cdot c_3(E) \cdot E^3 = \sqrt{GM_{\odot} \left(\frac{1 - e}{q} \right)^3} \cdot (t - t_0)$$

und mit Einführung einer neuen Größe

$$U = \sqrt{\frac{3e \cdot c_3(E)}{1 - e}} \cdot E$$

die modifizierte Form

$$U + \frac{1}{3}U^3 = \sqrt{6ec_3(E)} \cdot \sqrt{\frac{GM_{\odot}}{2q^3}} \cdot (t - t_0) .$$

Diese Gleichung hat bereits die formale Gestalt der Barkerschen Gleichung und kann bei bekannter rechter Seite nach U aufgelöst werden. Weiter erhält man mit $c_1(E) = \sin(E)/E$ und $c_2(E) = (1 - \cos(E))/E^2$

$$\begin{aligned} x &= r \cos \nu = \frac{q}{1 - e} (\cos E - e) = q \cdot \left(1 - \left[\frac{2c_2}{6ec_3} \right] U^2 \right) \\ y &= r \sin \nu = \frac{q}{1 - e} \sqrt{1 - e^2} \sin E = 2q \cdot \sqrt{\frac{1 + e}{2e}} \cdot \left[\frac{1}{6c_3} \right] \cdot c_1 \cdot U \quad (4.16) \\ r &= q \cdot \left(1 + \left[\frac{2c_2}{6c_3} \right] U^2 \right) . \end{aligned}$$

Später werden noch die Formeln für die Geschwindigkeit des Kometen benötigt, deren Ableitung allerdings etwas aufwendiger wäre und deshalb hier nicht gezeigt wird:

$$\begin{aligned} \dot{x} &= -\sqrt{\frac{GM_{\odot}}{q(1 + e)}} \left(\frac{y}{r} \right) \\ \dot{y} &= +\sqrt{\frac{GM_{\odot}}{q(1 + e)}} \left(\frac{x}{r} + e \right) . \end{aligned} \quad (4.17)$$

Zwischen U und der wahren Anomalie ν besteht ferner der Zusammenhang

$$\tan \left(\frac{\nu}{2} \right) = \left[\sqrt{\frac{1 + e}{3ec_3}} \cdot \frac{c_2}{c_1} \right] \cdot U ,$$

der hier der Vollständigkeit halber erwähnt ist.

Alle Terme, die in den obigen Gleichungen in eckige Klammern gesetzt sind, haben die Eigenschaft, daß sie für $E \rightarrow 0$ und $e \rightarrow 1$ den Wert Eins annehmen. In diesem Grenzfall gehen die Formeln in die bereits behandelten Gleichungen für Parabelbahnen über.

Die gezeigten Umformungen verhindern natürlich nicht, daß die Keplergleichung auch weiterhin iterativ gelöst werden muß. Wesentlich ist aber, daß die Iteration keine numerischen Schwierigkeiten mehr bereitet. Man beginnt zunächst mit der Annahme $E \approx 0$ ($c_3(E) \approx 1/6$) und bestimmt den zugehörigen Wert U durch Auflösung der Kepler-/Barkerschen Gleichung

$$U = B - 1/B$$

mit

$$B = \sqrt[3]{A + \sqrt{A^2 + 1}} \quad \text{und} \quad A = \frac{3}{2} \sqrt{6ec_3(E)} \sqrt{\frac{GM_\odot}{2q^3}} \cdot (t - t_0) .$$

Hieraus folgen verbesserte Werte

$$\hat{E} = U \cdot \sqrt{\frac{1-e}{3e \cdot c_3(E)}} \quad \text{und} \quad c_3(\hat{E}) = \frac{\hat{E} - \sin \hat{E}}{\hat{E}^2} ,$$

mit denen man wieder ein genaueres \hat{u} erhält. Diese Schritte werden wiederholt, bis sich U im Rahmen der geforderten Genauigkeit nicht mehr verändert. Wichtig ist dabei allerdings, daß man die Funktionen $c_1(E)$, $c_2(E)$ und $c_3(E)$ für kleine Werte von E richtig berechnet. Aufgrund gegenseitiger Auslöschung einzelner Terme ist dies mit einer direkten Auswertung der trigonometrischen Funktionen nicht möglich. Stattdessen verwendet man die Taylorentwicklungen von c_1 , c_2 und c_3 , die folgende Form haben:

$$\begin{aligned} c_1(E) &= \frac{\sin E}{E} = 1 - \frac{E^2}{3!} + \frac{E^4}{5!} - \dots = \sum_{n=1}^{\infty} \alpha_n \\ c_2(E) &= \frac{1 - \cos E}{E^2} = \frac{1}{2!} - \frac{E^2}{4!} + \frac{E^4}{6!} - \dots = \sum_{n=1}^{\infty} \beta_n \\ c_3(E) &= \frac{E - \sin E}{E^3} = \frac{1}{3!} - \frac{E^2}{5!} + \frac{E^4}{7!} - \dots = \sum_{n=1}^{\infty} \gamma_n . \end{aligned} \quad (4.18)$$

Alle drei Reihen enthalten ausschließlich gerade Potenzen von E und konvergieren daher sehr rasch. Die Summanden lassen sich bequem mit folgender Rekursion auswerten:

$$\begin{aligned} \alpha_1 &= 1 \\ \beta_n &= \alpha_n \cdot \frac{1}{2n}, \quad \gamma_n = \beta_n \cdot \frac{1}{2n+1}, \quad \alpha_{n+1} = -E^2 \cdot \gamma_n \quad (n = 1, \dots) . \end{aligned}$$

Sie wird in der Funktion **Stumpff** solange ausgeführt, bis ein Summand kleiner als die gewünschte Genauigkeit **eps** wird.

```
-----
// Stumpff: Auswertung der Stumpff'schen Funktionen C1, C2 and C3
// E2      Quadrat der exzentrischen Anomalie (E2=E*E) in [rad^2]
// c1      Wert von C1 = sin(E)/E
// c2      Wert von C2 = (1-cos(E))/(E*E)
// c3      Wert von C3 = (E-sin(E))/(E^3)
-----
void Stumpff (double E2, double& c1, double& c2, double& c3)
{
    const double eps = 100.0*eps_mach;
    double n, add;
    c1 = c2 = c3 = 0.0;
    add = n = 1.0;
    do {
        c1 += add; add /= (2.0*n);
        c2 += add; add /= (2.0*n+1.0);
        c3 += add; add *= -E2;
        n += 1.0;
    }
    while (fabs(add) >= eps);
}
```

Die übrigen Gleichungen für parabolische und parabelnahe Kometenbahnen sind in der Funktion **Parab** programmiert, die genauso aufgerufen wird wie **Hyperb**.

```
-----
// Parab: Berechnung der Orts- und Geschwindigkeitsvektoren fuer parabolische
//        und nahe-parabolische Bahnen
// GM      Produkt aus Gravitationskonstante und Zentralmasse [AE^3*d^-2]
// t0      Zeitpunkt des Periheldurchgangs
// t      Berechnungszeitpunkt
// q      Periheldistanz in [AE]
// e      Exzentrizitaet der Bahn (^1)
// r      Ortsvektor bezogen auf die Bahnebene in [AE]
// v      Geschwindigkeitsvektor bezogen auf die Bahnebene in [AE/d]
// Beachte: t0 und t in Julianischen Jahrhunderten seit J2000
// -----
void Parab ( double GM, double t0, double t, double q, double e,
             Vec3D& r, Vec3D& v )
{
    const int maxit = 15;
    const double eps = 100.0*eps_mach;
    int i=0;
    double E2=0.0;
    double E20, fac, c1, c2, c3, k, tau, A, B, u, u2, R;
    fac = 0.5*e;
    k = sqrt(GM / (q*(1.0+e)));
    tau = sqrt(GM)*(t-t0);
    do {
        ++i;
        E20 = E2;
        A = 1.5*sqrt(fac/(q*q*q))*tau;
        B = pow ( sqrt(A*A+1.0)+A, 1.0/3.0 );
        u = B - 1.0/B;
        u2 = u*u;
    }
```

```

E2 = u2*(1.0-e)/fac;
Stumpff (E2, c1,c2,c3);
fac = 3.0*e*c3;
if (i == maxit) { cerr << " Konvergenzprobleme in Parab" << endl; break; }
}
while (fabs(E2-E20) >= eps);
R = q * ( 1.0 + u2*c2*e/fac );
r = Vec3D (q*(1.0-u2*c2/fac), q*sqrt((1.0+e)/fac)*u*c1, 0.0);
v = Vec3D (-k*r[y]/R, k*(r[x]/R+e), 0.0);
}

```

4.5 Die Gaußschen Vektoren

Die bisher zusammengestellten Funktionen erlauben die Berechnung des Kometenortes innerhalb der Bahnebene. Um nun die ekliptikalnen Koordinaten des Kometen bestimmen zu können, muß man zunächst die Lage der Bahn relativ zur Ekliptik festlegen. Dazu dienen drei weitere Bahnelemente (Abb. 4.6):

- i* Die *Bahnneigung* gibt an, in welchem Winkel sich die Bahnebene und die Ekliptik schneiden. Ein Schnittwinkel von über 90° bedeutet, daß der Komet sich rückläufig bewegt, sein Umlaufssinn um die Sonne also dem der Planeten entgegengesetzt ist.
- Ω Die *Länge des aufsteigenden Knotens* bezeichnet den Winkel zwischen dem Frühlingspunkt und demjenigen Punkt der Bahn, in dem der Komet die Ekliptik von Süd nach Nord kreuzt.
- ω Das *Argument des Perihels* gibt den Winkel zwischen der Richtung des aufsteigenden Knotens und der Richtung zum sonnennächsten Punkt der Bahn an.

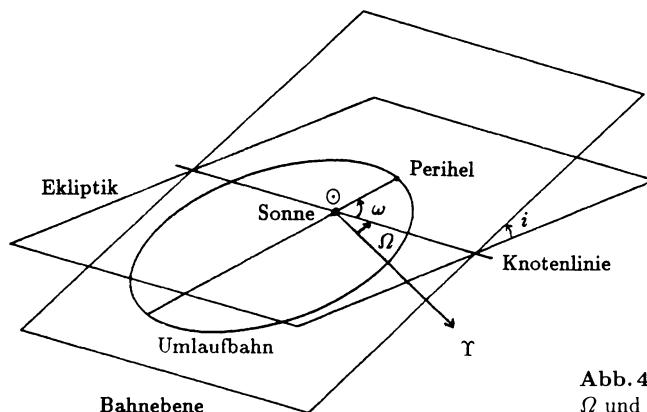


Abb. 4.6. Die Lagebahnelemente i , Ω und ω des Kometen

Anstelle des letztgenannten Elements ω wird sehr oft die *Länge des Perihels* ($\varpi = \Omega + \omega$) verwendet, die als Summe aus Knotenlänge und Argument des Perihels definiert ist.

Da sich die Lage von Frühlingspunkt und Ekliptik aufgrund der Präzession langsam verändert, wird zu den Bahnelementen auch deren Äquinoktium mit angegeben. „Äquinoktium 1950“ bedeutet zum Beispiel, daß sich die Elemente auf den Frühlingspunkt und die Ekliptik von 1950 beziehen.

Zur Ableitung der Transformationsgleichungen betrachtet man zunächst das perifokale Koordinatensystem, das parallel zur Bahnebene und zur Apsidenlinie ausgerichtet ist. Ein Bahnpunkt mit der wahren Anomalie ν und der Entfernung r von der Sonne hat darin die Koordinaten

$$\tilde{\mathbf{r}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \begin{pmatrix} r \cos \nu \\ r \sin \nu \\ 0 \end{pmatrix} ,$$

die zur weiteren Unterscheidung von den ekliptikalnen Koordinaten (x, y, z) durch eine Tilde gekennzeichnet sind.

Das perifokale Koordinatensystem geht durch eine positive Drehung mit Winkel ω um die \tilde{z}/z'' -Achse aus einem Koordinatensystem (x'', y'', z'') hervor, dessen x'' -Achse in Richtung des aufsteigenden Knotens weist und dessen $x''-y''$ -Ebene mit der Bahnebene zusammenfällt (Abb.4.7). Für die Koordinaten des Bahnpunktes in diesem System gilt entsprechend

$$\mathbf{r}'' = \mathbf{R}_z(-\omega) \tilde{\mathbf{r}} = \begin{pmatrix} +\cos \omega & -\sin \omega & 0 \\ +\sin \omega & +\cos \omega & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r \cos \nu \\ r \sin \nu \\ 0 \end{pmatrix} = \begin{pmatrix} r \cos u \\ r \sin u \\ 0 \end{pmatrix} ,$$

wobei das *Argument der Breite* u die Summe aus dem Argument des Perihels und der wahren Anomalie bezeichnet:

$$u = \omega + \nu . \quad (4.19)$$

Nun geht man über zum System (x', y', z') , dessen x' -Achse ebenfalls in Richtung der Knotenlinie zeigt, dessen $x'-y'$ -Ebene aber in der Ekliptik liegt. Beide Koordinatensysteme sind gerade um den Winkel i der Bahnneigung gegeneinander verdreht:

$$\mathbf{r}' = \mathbf{R}_x(-i) \mathbf{r}'' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & +\cos i & -\sin i \\ 0 & +\sin i & +\cos i \end{pmatrix} \begin{pmatrix} r \cos u \\ r \sin u \\ 0 \end{pmatrix} = \begin{pmatrix} r \cos u \\ r \sin u \cos i \\ r \sin u \sin i \end{pmatrix} .$$

Die so erhaltenen Koordinaten beziehen sich bereits auf die Ekliptik, werden allerdings noch vom aufsteigenden Knoten aus gezählt. Deshalb wird im folgenden Schritt noch einmal in ein neues Koordinatensystem gewechselt, dessen Achsen (x, y, z) in der gewohnten Weise auf Frühlingspunkt und Ekliptik ausgerichtet sind:

$$\mathbf{r} = \mathbf{R}_z(-\Omega) \mathbf{r}' = \begin{pmatrix} +\cos \Omega & -\sin \Omega & 0 \\ +\sin \Omega & +\cos \Omega & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r \cos u \\ r \sin u \cos i \\ r \sin u \sin i \end{pmatrix} .$$

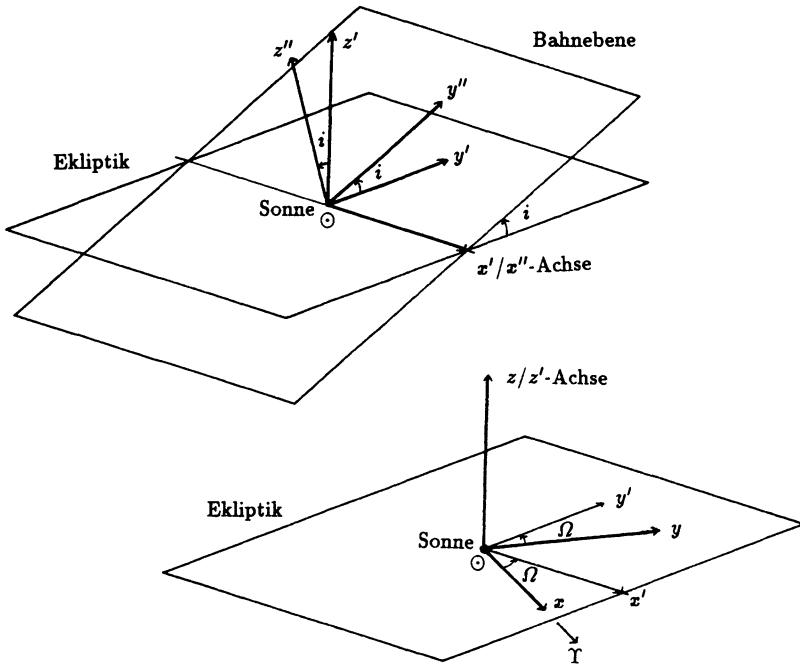


Abb. 4.7. Übergang zu ekliptikalnen Koordinaten

Zusammen erhält man so die Gleichungen

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = r \begin{pmatrix} \cos u \cos \Omega - \sin u \cos i \sin \Omega \\ \cos u \sin \Omega + \sin u \cos i \cos \Omega \\ \sin u \sin i \end{pmatrix}, \quad (4.20)$$

mit deren Hilfe sich die ekliptikalnen Koordinaten (x, y, z) eines gegebenen Bahn-punkts aus der Entfernung r und der wahren Anomalie ν berechnen lassen.

Alternativ dazu lassen sich die oben beschriebenen Drehungen auch in einer einzigen Matrix

$$\mathbf{U} = \mathbf{R}_z(-\Omega) \mathbf{R}_x(-i) \mathbf{R}_z(-\omega) \quad (4.21)$$

zusammenfassen, mit deren Hilfe sich die vollständige Transformation

$$\mathbf{r} = \mathbf{U} \hat{\mathbf{r}} \quad \dot{\mathbf{r}} = \mathbf{U} \dot{\hat{\mathbf{r}}} \quad (4.22)$$

zwischen perifokalen und ekliptikalnen Koordinaten besonders kompakt beschreiben lässt. Die Verwendung dieser Beziehungen ist zum Beispiel dann empfehlenswert, wenn man neben der Position auch die Geschwindigkeit betrachtet, oder wenn man eine ganze Reihe heliozentrischer Kometenorte auf einmal berechnen will. Die Matrixelemente brauchen dann nur einmal zu Beginn bestimmt werden, da sie nur von den Bahnelementen und nicht von der Zeit abhängen.

Die Spalten (\mathbf{P} , \mathbf{Q} , \mathbf{R}) der Matrix \mathbf{U} stellen drei Einheitsvektoren dar, die aufeinander senkrecht stehen und als Gaußsche Vektoren bezeichnet werden. Dabei liegen

$$\mathbf{P} = \begin{pmatrix} +\cos\omega\cos\Omega - \sin\omega\cos i \sin\Omega \\ +\cos\omega\sin\Omega + \sin\omega\cos i \cos\Omega \\ +\sin\omega\sin i \end{pmatrix} \quad (4.23)$$

und

$$\mathbf{Q} = \begin{pmatrix} -\sin\omega\cos\Omega - \cos\omega\cos i \sin\Omega \\ -\sin\omega\sin\Omega + \cos\omega\cos i \cos\Omega \\ +\cos\omega\sin i \end{pmatrix} \quad (4.24)$$

in der Bahnebene und weisen in die Richtung des Perihels beziehungsweise in Richtung eines Punktes mit der wahren Anomalie $\nu = 90^\circ$. Der dritte Vektor

$$\mathbf{R} = \begin{pmatrix} +\sin i \sin\Omega \\ -\sin i \cos\Omega \\ +\cos i \end{pmatrix} \quad (4.25)$$

schließlich steht senkrecht auf der Bahnebene und hängt somit lediglich von i und Ω , nicht aber vom Argument des Perihels ω ab.

Die Funktion `GaussVec` berechnet aus den Lageelementen der Bahn die Gaußschen Vektoren in Form einer 3×3 -Matrix der Klasse `Mat3D`. In Analogie zu (4.21) wird die gesamte Transformationsmatrix dabei aus drei Elementardrehungen aufgebaut. Gegenüber der komponentenweisen Darstellung der Gaußschen Vektoren erlaubt dies eine besonders kurze und übersichtliche Programmierung:

```
//-----
// GaussVec: Berechnet die Transformationsmatrix vom Koordinatensystem der
// Bahnebene in ekliptikale Koordinaten
// Omega Laenge des aufsteigenden Knotens der Bahn in [rad]
// i Neigung der Bahn zur Ekliptik in [rad]
// omega Argument des Perihels in [rad]
// <return> Transformationsmatrix mit den Gauss'schen Vektoren P, Q und R
//-----

Mat3D GaussVec (double Omega, double i, double omega)
{
    return R_z(-Omega) * R_x(-i) * R_z(-omega);
}
```

Die bisherigen entwickelten Programmteile zur Kometenbewegung lassen sich gut in einer Funktion zusammenfassen, die sich für alle Bahnformen eignet und direkt die ekliptikalnen heliozentrischen Koordinaten liefert.

```
//-----
// Kepler: Berechnung von Ort und Geschwindigkeit fuer Keplersche Bahnen
// bezogen auf die Ekliptik
// GM Produkt aus Gravitationskonstante und Zentralmasse [AE^3*d^-2]
// t0 Zeitpunkt des Periheldurchgangs
// t Berechnungszeitpunkt
// q Periheldistanz in [AE]
```

```

// e      Exzentrizitaet der Bahn
// PQR    Transformationsmatrix Bahnebene -> Ekliptik (Gauss'sche Vektoren)
// r      Heliozentrischer ekliptikal Ortsvektor in [AE]
// v      Heliozentrische ekliptikale Geschwindigkeit in [AE/d]
// Beachte: t0 und t in Julianischen Jahrhunderten seit J2000
//-----
void Kepler ( double GM, double t0, double t,
              double q, double e, const Mat3D& PQR,
              Vec3D& r, Vec3D& v )
{
    const double M0 = 0.1; // [rad]
    const double eps = 0.1;
    double M, delta, tau, invax;
    Vec3D r_orb,v_orb;
    delta = fabs(1.0-e);
    invax = delta / q;
    tau = sqrt(GM)*(t-t0);
    M = tau * sqrt(invax*invax*invax);
    if ( (M < M0) && (delta < eps) )
        Parab (GM, t0,t, q,e, r_orb, v_orb);
    else if ( e < 1.0 )
        Ellip (GM, M, 1.0/invax, e, r_orb, v_orb);
    else
        Hyperb (GM, t0,t, 1.0/invax, e, r_orb, v_orb);
    r = PQR*r_orb;
    v = PQR*v_orb;
}

```

4.6 Die Lichtlaufzeit

Das beobachtete Licht des Kometen benötigt für die Strecke zu uns eine Zeit, die je nach seiner Entfernung im Rahmen einiger Minuten bis Stunden liegen kann. Da sich der Komet während dieser Lichtlaufzeit weiterbewegt, liegt der beobachtete Ort ein wenig hinter dem tatsächlichen geometrischen Ort des Kometen zurück. Da die Lichtgeschwindigkeit c wesentlich größer ist als die typische Geschwindigkeit eines Kometen, bedeutet die Berücksichtigung der Lichtlaufzeit meist nur eine kleine Korrektur im Bereich von einigen Bogensekunden bis zu etwa einer Bogenminute. Man kann deshalb von einigen Näherungen Gebrauch machen, die es gestatten, die Lichtlaufzeit relativ einfach zu behandeln. Um den Ort $\mathbf{r}(t)$ zu berechnen, an dem der Komet zur Zeit t gesehen wird, berechnet man zunächst für diesen Zeitpunkt die heliozentrischen Koordinaten $\mathbf{r}_k(t)$ des Kometen und die geozentrischen Sonnenkoordinaten $\mathbf{r}_\odot(t)$. Die Strecke

$$\Delta_0 = |\mathbf{r}_0| = |\mathbf{r}_\odot(t) + \mathbf{r}_k(t)| = \sqrt{(x_\odot + x_k)^2 + (y_\odot + y_k)^2 + (z_\odot + z_k)^2}$$

gibt dann die geometrische Entfernung des Kometen von der Erde zur Zeit t an. Sie unterscheidet sich nur wenig von der vom Lichtstrahl tatsächlich zurückgelegten Strecke

$$\Delta = |\mathbf{r}_\odot(t) + \mathbf{r}_k(t')| \text{ mit } t' = t - \frac{\Delta}{c} ,$$

die aber nur iterativ berechnet werden kann, weil der Zeitpunkt t' der Lichtaussendung vom Kometen nicht von vornherein bekannt ist. Die Lichtlaufzeit ist damit $\tau \approx \Delta_0/c$ und der heliozentrische Kometenort zur Zeit t' ergibt sich in guter Näherung zu

$$\mathbf{r}_k(t') \approx \mathbf{r}_k(t) - \mathbf{v}_k(t) \cdot \frac{\Delta_0}{c} .$$

Mit den so *retardierten* (d.h. zurückversetzten) heliozentrischen Koordinaten des Kometen ist der beobachtete geozentrische Ortsvektor dann

$$\mathbf{r} \approx \mathbf{r}_\odot(t) + \left\{ \mathbf{r}_k(t) - \mathbf{v}_k(t) \cdot \frac{\Delta_0}{c} \right\} . \quad (4.26)$$

Der Faktor $1/c$ hat in den hier gebräuchlichen Einheiten AE und d den Wert

$$1/c = 0.00578/\text{AE} .$$

Die in der beschriebenen Weise korrigierten Koordinaten des Kometen werden als *astrometrische* Koordinaten bezeichnet. Sie eignen sich direkt für den Vergleich der Kometenposition mit Sternkoordinaten in einem Himmelsatlas. Anzumerken ist noch, daß für die geozentrische Entfernung üblicherweise nur der Wert der geometrischen Entfernung Δ_0 (und nicht der Lichtweg Δ) angegeben wird.

4.7 Das Programm COMET

Das Programm Comet berechnet Ephemeriden von Kometen und Asteroiden nach dem Zweikörperproblem. Die Störungen des betrachteten Himmelskörpers durch die anderen Planeten werden dabei nicht erfaßt. Diese Vereinfachung genügt jedoch meist für die Anforderungen der Praxis. Beim Vergleich der Ergebnisse dieses Programmes mit anderen Quellen sollte man aber darauf achten, ob die dort angegebenen Daten unter derselben Voraussetzung berechnet wurden. Der Benutzer muß sich nicht darum kümmern, welcher Bahntyp (Ellipse, Parabel oder Hyperbel) vorliegt. Comet erkennt dies anhand der gegebenen Exzentrizität und wählt intern den dazu passenden Rechenweg. Langgestreckte Ellipsen mit hoher Exzentrizität werden dabei nach dem numerisch sicheren Verfahren von Stumpff behandelt.

Comet berechnet heliozentrische ekliptikale und geozentrische äquatoriale Koordinaten. In beiden Fällen wird die Präzession auf das gewünschte Äquinoktium berücksichtigt. Die geozentrischen Koordinaten sind zusätzlich für den Einfluß der Lichtlaufzeit korrigiert. Die geozentrischen Koordinaten in dieser Form (mit Berücksichtigung von Präzession und Lichtlaufzeit) werden als astrometrische Koordinaten bezeichnet. Sie sind unmittelbar mit den Positionen in einem Sternkatalog vergleichbar oder man kann sie zum Aufsuchen des Objekts in eine Himmelskarte (desselben Äquinoktiums!) einzeichnen.

Bei Vergleichen mit anderen Quellen ist wiederum genau darauf zu achten, ob dort wirklich astrometrische Koordinaten im selben Äquinoktium angegeben sind, oder ob es sich dort eventuell um scheinbare Koordinaten (bei denen zusätzlich noch Nutation und Aberration berücksichtigt werden) handelt.

```

//-----
// Datei: Comet.cpp
// Zweck: Ephemeridenrechnung fuer Kometen und Asteroiden
//         (ungestoerter Zweikörperproblem fuer beliebige Exzentrizitaeten)
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//-----

#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>

#include "APC_Const.h"
#include "APC_IO.h"
#include "APC_Kepler.h"
#include "APC_Math.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"

using namespace std;

//-----
// GetElm: Einlesen der Bahnelemente aus einer Datei
//   Filename Name der Eingabedatei mit den Bahnelementen
//   Mjd0     Zeitpunkt des Periheldurchgangs als MJD
//   q        Periheldistanz in [AE]
//   e        Exzentrizitaet
//   PQR      Transformationsmatrix Bahnebene -> Ekliptik
//   T_eqx0   Aequinoktium der Bahnelemente
//-----
void GetElm ( char* Filename,
              double& Mjd0, double& q, double& e, Mat3D& PQR, double& T_eqx0 )
{
    int      year, month;
    double   Omega, i, omega;
    double   Year,day;
    ifstream inp;
    inp.open(Filename); // Datei zum Lesen oeffnen
    // Bahnelemente lesen und anzeigen
    cout << endl
        << " Bahnelemente aus der Datei " << Filename << endl
        << endl;
    inp >> year >> month >> day; inp.ignore(81,'\'n\'');
    Mjd0 = Mjd(year,month,int(day))+(day-int(day));
    cout << " Perihelzeit (j m t) "
        << " " << setprecision(2) << DateTime(Mjd0,DDd) << endl;
    inp >> q; inp.ignore(81,'\'n\'');
    cout << " Periheldistanz (q) "
        << setprecision(7) << setw(14) << q << " AE" << endl;
    inp >> e; inp.ignore(81,'\'n\'');
    cout << " Exzentrizitaet (e) "
        << setprecision(7) << setw(14) << e << endl;
}

```

```

inp >> i; inp.ignore(81,'\'n');
cout << " Bahneigung (i)           "
     << setprecision(5) << setw(12) << i << " Grad" << endl;
inp >> Omega; inp.ignore(81,'\'n');
cout << " Knotenlaenge          "
     << setprecision(5) << setw(12) << Omega << " Grad" << endl;
inp >> omega; inp.ignore(81,'\'n');
cout << " Argument des Perihels "
     << setprecision(5) << setw(12) << omega << " Grad" << endl;
inp >> Year; inp.ignore(81,'\'n');
cout << " Aequinoktium          "
     << setprecision(2) << setw(9)  << Year << endl;
inp.close();
T_eqx0 = (Year-2000.0)/100.0;
PQR = GaussVec(Rad*Omega, Rad*i, Rad*omega);
}

//-----
// GetEph: Eingabe des Zeitraums der Ephemeride, der Schrittweite und des
//         Aequinoktiums
// MjdStart Anfangsdatum der Ephemeride als MJD
// Step      Schrittweite der Ephemeride in [d]
// MjdEnd    Enddatum der Ephemeride als MJD
// T_eqx     Gewuenschtes Aequinoktium der Ephemeride in Julianischen
//           Jahrhunderten seit J2000
//-----
void GetEph (double& MjdStart, double& Step, double& MjdEnd, double& T_eqx)
{
    int      year, month, day;
    double   hour, Year;
    cout << endl
        << " Beginn und Ende der Ephemeride: " << endl
        << endl;
    // Abfrage von Beginn und Ende, Schrittweite und Aequinoktium
    cout << " Erstes Berechnungsdatum (JJJJ MM TT HH.HHH) ... ";
    cin >> year >> month >> day >> hour; cin.ignore(81,'\'n');
    MjdStart = Mjd(year, month, day) + hour/24.0 ;
    cout << " Letztes Berechnungsdatum (JJJJ MM TT HH.HHH) ... ";
    cin >> year >> month >> day >> hour; cin.ignore(81,'\'n');
    MjdEnd = Mjd(year, month, day) + hour/24.0 ;
    cout << " Schrittweite (TT HH.HH)           ... ";
    cin >> day >> hour; cin.ignore(81,'\'n');
    Step  = day + hour/24.0;
    cout << endl
        << " Gewuenschtes Aequinoktium der Ephemeride (JJJJ) ... ";
    cin >> Year; cin.ignore(81,'\'n');
    T_eqx = (Year-2000.0)/100.0;
}

//-----
// Hauptprogramm
//-----
void main(int argc, char* argv[]) {

```

```

// Variablen
int      n_line = 0;
double   MjdStart, Step, MjdEnd, T_eqx;
double   Mjd0, q, e, T_eqx0;
Mat3D    PQR;
double   Date, T;
double   dist, fac;
Vec3D   R_Sun, r_helioc, v_helioc, r_geoc, r_eq;
char     InputFile[APC_MaxFilename] = "";
char     OutputFile[APC_MaxFilename] = "";
bool    FoundInputfile = false;
bool    FoundOutputfile = false;
ofstream OutFile;

// Titel
cout << endl
    << " COMET: Ephemeridenrechnung fuer Kometen und Kleinplaneten" << endl
    << "           (c) 1999 Oliver Montenbruck, Thomas Pfleger      " << endl
    << endl;

// Ermittle die Namen der Ein- und Ausgabedatei
GetFilenames( argc, argv, "Comet.dat", InputFile, FoundInputfile,
               OutputFile, FoundOutputfile );
// Abbruch, falls die Eingabedatei nicht gefunden wurde
if ( !FoundInputfile ) {
    cerr << " Abbruch." << endl;
    exit(-1);
}

// Bahnelemente lesen, Zeitraum und Aequinoktium eingeben
GetElm ( InputFile, Mjd0, q, e, PQR, T_eqx0 );
GetEph ( MjdStart, Step, MjdEnd, T_eqx );
PQR = PrecMatrix_Ecl(T_eqx0, T_eqx) * PQR;

// Bei Bedarf die Ausgabe in eine Datei umlenken
if (FoundOutputfile) {
    OutFile.open(OutputFile);
    if (OutFile.is_open()) cout = OutFile;
}

// Vorspann
cout << endl << endl
    << "    Datum      ET  Sonne    l      b      r"
    << "        RA          Dec      Entfernung " << endl
    << setw(43) << " " << " h  m  s      o  ' \r"      (AE) " << endl;

// Ephemeride berechnen
Date = MjdStart;

// Schleife ueber Berechnungszeitpunkte
while ( Date < MjdEnd + Step/2 ) {

    // Geozentrische ekliptikale Koordinaten der Sonne, Aequinoktium T_eqx
    T = ( Date - 51544.5 ) / 36525.0;
    R_Sun = PrecMatrix_Ecl(T, T_eqx) * SunPos(T);
}

```

```

// Heliozentrische ekliptikale Koordinaten des Körpers, Äquinoxtium T_eqx
Kepler (GM_Sun, Mjd0, Date, q, e, PQR, r_helioc, v_helioc);

// Geometrische geozentrische Koordinaten des Körpers
r_geoc = r_helioc + R_Sun;

// Korrektur (1. Ordnung) für die Lichtlaufzeit
dist = Norm(r_geoc);
fac = 0.00578*dist;
r_geoc = r_geoc - fac*v_helioc;

// Äquatoriale Koordinaten
r_eq = Ecl2EquMatrix(T_eqx) * r_geoc;

// Ausgabe
cout << DateTime(Date,HHh)
    << fixed << setprecision(1)
    << setw(7) << Deg*R_Sun[phi]
    << setw(7) << Deg*r_helioc[phi]
    << setw(6) << Deg*r_helioc[theta]
    << setprecision(3) << setw(7) << r_helioc[r]
    << setprecision(1) << setw(12) << Angle(Deg*r_eq[phi]/15.0,DMMSSs)
    << " " << showpos << setw(9) << Angle(Deg*r_eq[theta],DMMSS)
    << noshowpos << setprecision(6) << setw(11) << dist
    << endl;
++n_line;
if ( (n_line % 5) == 0 ) cout << endl; // Leerzeile alle 5 Zeilen

Date += Step; // Nächster Berechnungszeitpunkt
};

if (OutFile.is_open()) OutFile.close();
}

```

Nun wollen wir als Beispiel für die Bedienung von Comet eine Ephemeride des Halleyschen Kometen für den Zeitraum seiner letzten Sichtbarkeit berechnen lassen. Zunächst sind die Bahnelemente in einer Datei mit dem Namen Comet.dat bereitzustellen. Die Kommentare (Ausrufezeichen einschließlich der Erklärungen dahinter) dienen nur der Dokumentation und können auf Wunsch auch fortgelassen werden. Für Halley bei seiner Sichtbarkeit 1985/86 sieht die Datei dann so aus:

```

1986 2 9.43867 ! Zeitpunkt des Periheldurchgangs (JJJ MM TT.T)
  0.5870992 ! Periheldistanz q in [AE]
  0.9672725 ! Exzentrizität e
  162.23932 ! Bahneigung i in [Grad]
   58.14397 ! Länge des aufsteigenden Knotens in [Grad]
  111.84658 ! Argument des Perihels in [Grad]
  1950.0      ! Äquinoxtium der Bahnelemente

```

Nach dem Start des Programms gibt Comet zunächst als Kontrolle die aus der Eingabedatei gelesenen Bahnelemente aus, bevor der Benutzer das Zeitintervall, die Schrittweite und das gewünschte Äquinoxtium einzugeben hat:

COMET: Ephemeridenrechnung fuer Kometen und Kleinplaneten
 (c) 1999 Oliver Montenbruck, Thomas Pfleger

Standard-Eingabedatei Comet.dat

Bahnelemente aus der Datei Comet.dat

Perihelzeit (j m t)	1986/02/09.44
Periheldistanz (q)	0.5870992 AE
Exzentritzaet (e)	0.9672725
Bahnneigung (i)	162.23932 Grad
Knotenlaenge	58.14397 Grad
Argument des Perihels	111.84658 Grad
Aequinoktium	1950.00

Wir wünschen eine Ephemeride für den Zeitraum vom 15. November 1985 bis zum 1. April 1986 bei einer Schrittweite von 10 Tagen. Als Äquinoktium wählen wir das Jahr 2000. Die Eckdaten der Ephemeride sind in der Form „Jahr, Monat, Tag und Stunde mit Dezimalanteil“ einzugeben. Die Schrittweite erwartet Comet im Format „Tage und Stunden mit Dezimalanteil“. Eingaben des Benutzers sind durch kursive Schrift gekennzeichnet.

Beginn und Ende der Ephemeride:

Erstes Berechnungsdatum (JJJJ MM TT HH.HHH)	... 1985 11 15 00.0
Letztes Berechnungsdatum (JJJJ MM TT HH.HHH)	... 1986 04 01 00.0
Schrittweite (TT HH.HH)	... 10 00.0

Gewünschtes Aequinoktium der Ephemeride (JJJJ) ... 2000.0

Nun berechnet Comet die Ephemeride nach den Vorgaben:

Datum	ET	Sonne	l	b	r	RA			Dec	Entfernung
						h	m	s		(AE)
1985/11/15 00.0	232.8	56.9	0.6	1.720	4 00 40.3	+22	04	27	0.736822	
1985/11/25 00.0	242.9	53.2	1.8	1.572	2 15 19.1	+18	24	11	0.623053	
1985/12/05 00.0	253.0	48.7	3.2	1.422	0 28 47.5	+10	39	00	0.665665	
1985/12/15 00.0	263.2	43.1	5.0	1.269	23 18 18.6	+ 3	53	22	0.821652	
1985/12/25 00.0	273.3	35.9	7.1	1.114	22 36 56.1	- 0	20	05	1.019722	
1986/01/04 00.0	283.5	26.2	9.8	0.961	22 10 36.0	- 3	00	26	1.216950	
1986/01/14 00.0	293.7	12.8	13.0	0.814	21 50 58.9	- 4	58	44	1.388260	
1986/01/24 00.0	303.9	353.4	16.2	0.687	21 33 26.2	- 6	47	36	1.511970	
1986/02/03 00.0	314.1	326.3	17.7	0.604	21 15 33.4	- 8	49	26	1.563479	
1986/02/13 00.0	324.2	294.8	14.9	0.592	20 57 11.7	-11	15	58	1.520207	
1986/02/23 00.0	334.3	267.2	8.7	0.658	20 39 33.7	-14	09	18	1.383204	
1986/03/05 00.0	344.3	247.1	2.6	0.775	20 22 12.2	-17	39	45	1.179039	
1986/03/15 00.0	354.3	232.8	-1.9	0.918	20 00 46.1	-22	28	16	0.936979	
1986/03/25 00.0	4.2	222.5	-5.2	1.070	19 22 34.1	-30	13	41	0.685469	
1986/04/04 00.0	14.1	214.7	-7.5	1.225	17 41 16.2	-42	56	51	0.475274	

Die einzelnen Spalten enthalten zunächst Datum und Uhrzeit. Darauf folgt in der Spalte **Sonne** die geozentrische ekliptikale Länge der Sonne. Mit Hilfe dieser Angabe kann man abschätzen, wie weit der Komet (oder Asteroid) am Himmel von der Sonne entfernt steht. Das liefert einen ersten Anhalt dafür, ob das Objekt der Beobachtung zugänglich ist. Die drei folgenden (mit **l**, **b** und **r** betitelten) Spalten enthalten die heliozentrischen ekliptikalnen Koordinaten des Himmelskörpers. In den letzten drei Spalten finden wir die geozentrischen Koordinaten Rektaszension, Deklination und die geozentrische geometrische (nicht für die Lichtlaufzeit korrigierte) Entfernung des Kometen in astronomischen Einheiten.

5. Störungsrechnung

Die Berechnung von Kometen- und Planetenbahnen auf der Grundlage des ungestörten Zweikörperproblems bietet eine einfache und überschaubare Möglichkeit, den Ort eines solchen Himmelskörpers zu beliebigen Zeiten vorherzusagen. Die analytische Darstellung der Bewegung mit Hilfe von Kegelschnitten vermittelt darüberhinaus auf besonders anschauliche Weise den Zusammenhang zwischen den verschiedenen Bahnelementen und dem Ort in der Bahn.

Demgegenüber steht jedoch der Nachteil, daß die von den großen Planeten ausgeübten Störungen nicht berücksichtigt werden können. Obwohl die Anziehungskräfte der Planeten im allgemeinen um einige Größenordnungen kleiner sind als die der Sonne, führen sie im Laufe mehrerer Jahre doch oft zu merklichen Abweichungen von einer ungestörten Bahn. Besonders ausgeprägt sind diese Störungen dann, wenn ein Komet oder Kleinplanet wiederholt nahe an einem der großen Planeten vorbeifliegt. In diesem Fall sind merkliche Änderungen möglich, wie man sie bei verschiedenen periodischen Kometen beobachten kann. Beispiele hierfür sind der Komet P/Pons-Winnecke (1819 III), dessen Bahnneigung durch die Nähe des Aphels zu Jupiter zwischen den Jahren 1819 und 1976 von $10^\circ 8'$ auf $22^\circ 3'$ angewachsen ist oder der Komet P/Wolf (1884 III), dessen Umlaufszeit sich 1922 in Folge einer Begegnung mit Jupiter von 6.8 Jahren auf 8.3 Jahre erhöhte.

Auch wenn die Störungen der großen Planeten für die meisten Asteroiden weit kleiner als in diesen Beispielen sind, so verhindern sie doch meist eine befriedigende Vorhersage der Bahn über mehrere Jahre auf der Grundlage einfacher Keplerbahnen. Für eine sichere Auffindung und Identifikation schwächer Kleinplaneten empfiehlt es sich deshalb, die genannten Störungen bei der Berechnung ihrer Ephemeriden zu berücksichtigen. Prinzipiell lassen sich hierzu zwei verschiedene Verfahren verwenden. Im Rahmen der *speziellen Störungsrechnung* berechnet man ausgehend vom Ort und der Geschwindigkeit des Planetoiden zu einer Anfangs-epochen den weiteren Bahnverlauf Schritt für Schritt mit Hilfe der numerischen Integration unter Berücksichtigung aller von Sonne und Planeten ausgeübten Beschleunigungen bis zu einem gewünschten Endzeitpunkt. Die zweite Möglichkeit der *allgemeinen Störungsrechnung* besteht darin, die Abweichungen von der ungestörten Bahn durch eine periodische Reihe darzustellen, die es ermöglicht, die gestörte Position zu beliebigen Zeiten zu berechnen. Diesem Vorteil einer analytischen Methode steht jedoch ein enormer Aufwand für die Aufstellung der Reihenentwicklung gegenüber, der eine Anwendung auf mehrere Tausend Kleinplaneten verhindert. Analytische Störungstheorien sind deshalb im wesentlichen nur für die großen Planeten verfügbar (vgl. Kap. 6).

5.1 Bewegungsgleichung

Die Grundlage für die Behandlung einer Kleinplanetenbahn mit Hilfe der numerischen Integration bildet die sogenannte Bewegungsgleichung, die die wirksame Beschleunigung als Funktion von Ort und Zeit darstellt. Auf einen Körper der Masse m im alleinigen Schwerkraftfeld der Sonne wirkt nach dem Newtonschen Gravitationsgesetz eine Kraft der Stärke

$$F = ma = GM_{\odot}m \frac{1}{r^2}, \quad (5.1)$$

die mit zunehmender Entfernung r von der Sonne quadratisch abnimmt. Hierin bezeichnet

$$GM_{\odot} = k^2 AE^3 d^{-2}$$

mit

$$k = 0.01720209895$$

wieder das Produkt aus Gravitationskonstante und Sonnenmasse (vgl. Kap. 4). Berücksichtigt man, daß die Kraft immer in Richtung auf die Sonne zu zeigt, dann ergibt sich die resultierende Beschleunigung mit Hilfe des entsprechenden Einheitsvektors $-\mathbf{r}/r$ zu

$$\mathbf{a} = -GM_{\odot} \frac{\mathbf{r}}{r^3}. \quad (5.2)$$

Wie man sieht hängt die Beschleunigung nicht von der Masse m des Körpers ab, da die Gravitationskraft selbst proportional zu m anwächst.

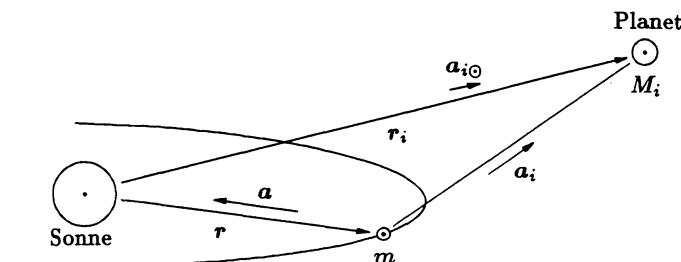


Abb. 5.1. Beschleunigung des Körpers m durch Sonne und Planeten

In entsprechender Weise führt die Anziehungschaft eines Planeten der Masse M_i am Ort \mathbf{r}_i (vgl. Abb. 5.1) zu einer zusätzlichen Beschleunigung

$$\mathbf{a}_i = -GM_i \frac{\mathbf{r} - \mathbf{r}_i}{|\mathbf{r} - \mathbf{r}_i|^3} \quad (5.3)$$

des betrachteten Körpers. Da der Planet jedoch gleichzeitig auch eine Beschleunigung der Stärke

$$\mathbf{a}_{i\odot} = -GM_i \frac{-\mathbf{r}_i}{|\mathbf{r}_i|^3} \quad (5.4)$$

auf die Sonne selbst ausübt, wirkt auf die Masse m insgesamt die Beschleunigung

$$\Delta\mathbf{a}_i = \mathbf{a}_i - \mathbf{a}_{i\odot} = -GM_i \left(\frac{\mathbf{r} - \mathbf{r}_i}{|\mathbf{r} - \mathbf{r}_i|^3} + \frac{\mathbf{r}_i}{|\mathbf{r}_i|^3} \right) \quad (5.5)$$

relativ zur Sonne. Der erste Term wird dabei auch als direkte, der zweite als indirekte Störbeschleunigung bezeichnet.

Faßt man die Beiträge aller neun Planeten zusammen, dann ergibt sich insgesamt die folgende Bewegungsgleichung für einen Kometen oder Kleinplaneten:

$$\mathbf{a} = -GM_{\odot} \frac{\mathbf{r}}{r^3} - \sum_{i=1}^9 GM_i \left(\frac{\mathbf{r} - \mathbf{r}_i}{|\mathbf{r} - \mathbf{r}_i|^3} + \frac{\mathbf{r}_i}{|\mathbf{r}_i|^3} \right) \quad . \quad (5.6)$$

Betrachtet man die in Tabelle 5.1 wiedergegebenen Verhältnisse von Sonnen- und Planetenmassen, dann erkennt man, daß selbst die größten Planeten Jupiter und Saturn im allgemeinen nur sehr kleine Störungen ausüben. Beträgt die Entfernung zu diesen Planeten jedoch weniger als 1/10 AE oder gar 1/100 AE, dann kann die Störbeschleunigung durchaus von der gleichen Größenordnung wie die Anziehung der Sonne sein.

Tabelle 5.1. Verhältnisse von Sonnenmasse (M_{\odot}) und Planetenmassen (IAU 1976)

Planet	Massenverhältnis
Merkur	$M_{\odot}/M_1 = 6023600.0$
Venus	$M_{\odot}/M_2 = 408523.5$
Erde/Mond	$M_{\odot}/M_3 = 328900.5$
Mars	$M_{\odot}/M_4 = 3098710.0$
Jupiter	$M_{\odot}/M_5 = 1047.355$
Saturn	$M_{\odot}/M_6 = 3498.5$
Uranus	$M_{\odot}/M_7 = 22869.0$
Neptun	$M_{\odot}/M_8 = 19314.0$
Pluto	$M_{\odot}/M_9 = 3000000.0$

Mit Hilfe der Funktion `Accel` kann die Beschleunigung $\mathbf{a} = (a_x, a_y, a_z)$ auf einen Körper mit den heliozentrischen Koordinaten $\mathbf{r} = (x, y, z)$ zu einem Zeitpunkt $T = (\text{MJD} - 51544.5)/36525$ berechnet werden. Voraussetzung hierfür ist, daß eine geeignete Funktion der Form

```
Vec3D Position (PlanetType Planet, double T);
```

mit

```
enum PlanetType { Sun, Mercury, Venus, Earth, Mars,
                  Jupiter, Saturn, Uranus, Neptune, Pluto };
```

zur Bereitstellung der entsprechenden heliozentrischen Planetenkoordinaten zur Verfügung steht. Das Koordinatensystem ist prinzipiell frei wählbar, im folgenden wird jedoch davon ausgegangen, daß sich die berechneten Koordinaten zunächst auf die Ekliptik und den Frühlingspunkt des Datums beziehen. Accel transformiert diese anschließend selbst in das System der Ekliptik und des Frühlingspunktes zur Epoche J2000, in dem die Bewegungsgleichung formuliert wird.

```
-----
// Accel: Berechnet den Beschleunigungsvektor fuer einen kleinen Koerper im
// Sonnensystem
// Mjd      Berechnungszeit als Modifiziertes Julianisches Datum
// r        Ekliptikale Koordinaten des Koerpers bezogen auf J2000
// <return> Beschleunigung in ekliptikalnen Koordinaten bezogen auf J2000
//          in [AE/d**2]
-----
Vec3D Accel (double Mjd, const Vec3D& r)
{
    // Grav.konstante * Sonnen- und Planetenmasse in AE**3/d**2
    static const double GM[10] = {
        GM_Sun,                      // Sonne
        GM_Sun / 6023600.0,           // Merkur
        GM_Sun / 408523.5,           // Venus
        GM_Sun / 328900.5,           // Erde
        GM_Sun / 3098710.0,          // Mars
        GM_Sun / 1047.355,           // Jupiter
        GM_Sun / 3498.5,             // Saturn
        GM_Sun / 22869.0,            // Uranus
        GM_Sun / 19314.0,            // Neptun
        GM_Sun / 3000000.0           // Pluto
    };
    // Variablen
    int iPlanet;
    PlanetType Planet;
    Vec3D a, r_p, d;
    Mat3D P;
    double T, D;
    // Anziehung durch die Sonne
    D = Norm(r); a = - GM_Sun * r / (D*D*D);
    // Stoerung der Planeten
    T = (Mjd - MJD_J2000) / 36525.0;
    P = PrecMatrix_Ecl (T, T_J2000); // Praezession
    for (iPlanet=Mercury; iPlanet<=Neptune; iPlanet++) {
        Planet = (PlanetType) iPlanet;
        // Planetenposition (Ekliptik und Aequinoktium J2000)
        // r_p = P * KepPosition(Planet,T); // Schnell, aber geringe Genauigkeit
        r_p = P * PertPosition(Planet,T); // Genauer, aber langsamer
        d = r - r_p;
        // Direkte Beschleunigung
        D = Norm(d); a += - GM[iPlanet] * d / (D*D*D);
        // Indirekte Beschleunigung
        D = Norm(r_p); a += - GM[iPlanet] * r_p / (D*D*D);
    }
    return a;
}
```

Tabelle 5.2. Bahnelemente der Planeten zur Epoche 2000 Januar 1.5 (JD 2451545.0) bezogen auf die Ekliptik und den Frühlingspunkt von J2000.

Planet	a (AE)	e	M	n	Ω	i	ϖ
Merkur	0.387099	0.205634	174°794	149472°6738/cy	48°331	7°004	77°455
Venus	0.723332	0.006773	50°407	58517°8149/cy	76°680	3°394	131°571
Erde	1.000000	0.016709	357°525	35999°3720/cy	174°876	0°000	102°940
Mars	1.523692	0.093405	19°387	19140°3023/cy	49°557	1°849	336°059
Jupiter	5.204267	0.048775	18°819	3033°6272/cy	100°491	1°305	15°558
Saturn	9.582018	0.055723	320°348	1213°8664/cy	113°643	2°485	89°657
Uranus	19.229412	0.044406	142°956	426°9282/cy	73°989	0°773	170°531
Neptun	30.103658	0.011214	267°765	217°9599/cy	131°794	1°768	37°444
Pluto	39.264230	0.244672	15°023	146°3183/cy	110°287	17°151	224°050

5.2 Planetenkoordinaten

Um zu einem gegebenen Zeitpunkt die Beschleunigung nach (5.6) berechnen zu können, benötigt man neben den heliozentrischen Koordinaten r des betrachteten Körpers auch die entsprechenden Koordinaten r_i der neun Planeten Merkur bis Pluto. Für hochgenaue Berechnungen kann man dazu auf im voraus erstellte Planetenephemeriden zurückgreifen, wie sie etwa vom Jet Propulsion Laboratory der NASA für wissenschaftliche Aufgaben zur Verfügung gestellt werden. Dem Vorteil der hohen Genauigkeit solcher Ephemeriden steht jedoch der Nachteil ihres großen Umfangs (bis zu 10 000 Koeffizienten pro Jahr) gegenüber. Glücklicherweise reichen in vielen Fällen aber bereits wesentlich einfachere, genäherte Ephemeriden aus, um die Störungen auf eine Kometen- oder Kleinplanetenbahn über mehrere Jahre hinweg mit einer Genauigkeit im Bogensekundenbereich zu berücksichtigen. Sieht man von nahen Begegnungen einmal ab, dann ist die Störbeschleunigung selbst klein gegen die von der Sonne ausgeübte Beschleunigung. Kleine Fehler in den Planetenkoordinaten wirken sich demnach üblicherweise nur als vernachlässigbare Fehler zweiter Ordnung in den resultierenden Koordinaten des gestörten Körpers aus.

Als einfachste Möglichkeit bietet sich deshalb an, die Koordinaten der Planeten selbst durch Keplerbahnen zu approximieren. Die damit erreichbare Genauigkeit ist allerdings insbesondere für die äußeren Planeten begrenzt und liegt beispielsweise bei Vorhersagen über zehn Jahre im Bereich von einigen Bogenminuten. Ein geeigneter Satz von Bahnelementen für den Zeitraum von 1995 bis 2005 ist in Tabelle 5.2 wiedergegeben. Er wird in den Funktionen `State` und `KepPosition` verwendet, um für alle Planeten genäherte Positionen (und Geschwindigkeiten) zu berechnen.

```
//-----
// State: Berechnet Ort und Geschwindigkeit ausgehend von Zweikörperbahnen
//          (wird von KepPosition() und KepVelocity() benutzt)
//  Planet  Identifiziert den Planeten
//  T      Zeit in Julianischen Jahrhunderten seit J2000
//  r      Heliozentrischer Ort [AE], Ekliptik und Aequinoktium des Datums
```



```

//-----
// KepPosition: Berechnung des Planetenortes aus Keplerschen Elementen
//   Planet      Identifiziert den Planeten
//   T          Zeit in Julianischen Jahrhunderten seit J2000
//   <return>  Heliozentrischer Ort [AE], Ekliptik und Aequinoktium des Datums
//-----

Vec3D KepPosition (PlanetType Planet, double T)
{
    Vec3D r, v;
    State ( Planet, T, r, v );
    return r;
}

```

Als Alternative zur Funktion `KepPosition` kann – wie oben angedeutet – auch die Funktion `PertPosition` eingesetzt werden, um in `Accel` die Positionen der störenden Planeten zu berechnen. `PertPosition` basiert auf den in Kap. 6 beschriebenen analytischen Reihenentwicklungen und liefert speziell für die äußeren Planeten wesentlich genauere Ergebnisse als `KepPosition`. Dieser Vorteil wird allerdings durch deutlich höhere Rechenzeiten erkauft.

5.3 Numerische Integration

Für die weitere Betrachtung werden der Ort \mathbf{r} und die Geschwindigkeit \mathbf{v} des betrachteten Körpers zur Zeit t in einem einzigen, sechsdimensionalen Vektor

$$\mathbf{y}(t) = \begin{pmatrix} \mathbf{r}(t) \\ \mathbf{v}(t) \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \\ \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{pmatrix} \quad (5.7)$$

zusammengefaßt, der als Zustandsvektor bezeichnet wird. Die zeitliche Änderung $\dot{\mathbf{y}}$ von Ort und Geschwindigkeit läßt sich damit durch eine sogenannte Differentiagleichung der Form

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}) \quad (5.8)$$

beschreiben, wobei die Funktion

$$\mathbf{f}(t, \mathbf{y}) = \begin{pmatrix} \mathbf{v}(t) \\ \mathbf{a}(t, \mathbf{r}) \end{pmatrix} \quad (5.9)$$

die Geschwindigkeit und die Beschleunigung des Körpers zusammenfaßt, die ihrerseits von Zeit und Ort abhängen.

Der Grundgedanke der numerischen Integration läßt sich damit sehr einfach formulieren. Kennt man zu einem Zeitpunkt t_0 den Zustandsvektor \mathbf{y}_0 , dann kann man daraus den ungefähren Zustandsvektor zu einer späteren Zeit $t_1 = t_0 + h$ über die Näherung

$$\mathbf{y}(t_1) \approx \mathbf{y}_1 = \mathbf{y}(t_0) + h \cdot \dot{\mathbf{y}}(t_0) = \mathbf{y}_0 + h \cdot \mathbf{f}(t_0, \mathbf{y}_0) \quad (5.10)$$

berechnen. Setzt man dieses Verfahren fort, dann erhält man aus t_1 und \mathbf{y}_1 zunächst eine Näherung des Zustandsvektors zur Zeit $t_2 = t_0 + 2h$ und in ganz entsprechender Weise genäherete Werte für Ort und Geschwindigkeit zu weiteren Zeitpunkten $t_i = t_0 + ih$. Für den praktischen Einsatz ist diese einfache Vorgehensweise allerdings weniger geeignet, da sie nur für sehr kleine Schrittweiten h genaue Ergebnisse liefert und für die Berechnung eines Umlaufs damit sehr viele Schritte erforderlich sind.

Um auch bei größeren Schrittweiten zu brauchbaren Ergebnissen zu kommen, verwendet man deshalb bei den sogenannten Mehrschrittverfahren verbesserte Näherungen, die auf mehreren vorhergehenden Werten von \mathbf{f} beruhen. Zur Illustration des Prinzips dieses Verfahrens sei angenommen, daß Näherungen \mathbf{y}_j für den Zustandsvektor $\mathbf{y}(t_j)$ zu den Zeiten $t_j = t_0 + jh$ mit $j = 0, 1, \dots, i$ bekannt seien. Integriert man beide Seiten von

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}) \quad (5.11)$$

bezüglich der Zeit t von t_i bis t_{i+1} , dann erhält man die äquivalente Beziehung

$$\mathbf{y}(t_{i+1}) = \mathbf{y}(t_i) + \int_{t_i}^{t_{i+1}} \mathbf{f}(t, \mathbf{y}(t)) dt . \quad (5.12)$$

Das Integral kann in dieser Form jedoch nicht unmittelbar berechnet werden, da es selbst vom unbekannten Zustandsvektor $\mathbf{y}(t)$ abhängt. Um dieses Problem zu umgehen, ersetzt man den Integranden durch ein Polynom $p(t)$, das einige der Funktionswerte

$$\mathbf{f}_j = \mathbf{f}(t_j, \mathbf{y}_j) \quad (5.13)$$

zu früheren Zeiten t_j interpoliert, die gemäß der obigen Annahme bereits bekannt sind.

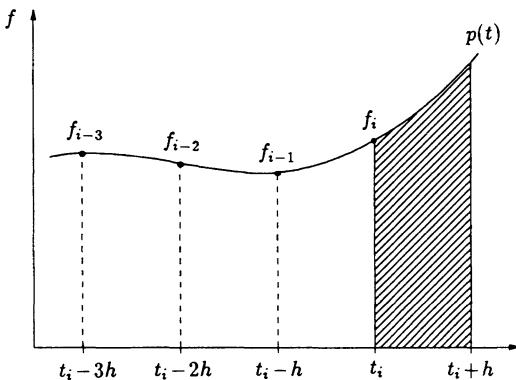


Abb. 5.2. Interpolation zurückliegender Funktionswerte und Integration zum nächsten Zeitschritt

Als Beispiel ist in Abb. 5.2 ein Polynom 3. Ordnung dargestellt, das durch vier Funktionswerte \mathbf{f}_{i-3} , \mathbf{f}_{i-2} , \mathbf{f}_{i-1} und \mathbf{f}_i zu den Zeiten t_{i-3} , t_{i-2} , t_{i-1} und t_i festgelegt ist. Schreibt man dieses Polynom in der Form

$$\mathbf{p}(t) = \mathbf{a}_0 + \mathbf{a}_1 \sigma + \mathbf{a}_2 \sigma^2 + \mathbf{a}_3 \sigma^3 \quad \text{mit} \quad \sigma = \frac{1}{h}(t - t_i) \quad (5.14)$$

und setzt die zugehörigen Koeffizienten

$$\begin{aligned} \mathbf{a}_0 &= \mathbf{f}_i \\ \mathbf{a}_1 &= (-2\mathbf{f}_{i-3} + 9\mathbf{f}_{i-2} - 18\mathbf{f}_{i-1} + 11\mathbf{f}_i)/6 \\ \mathbf{a}_2 &= (-3\mathbf{f}_{i-3} + 12\mathbf{f}_{i-2} - 15\mathbf{f}_{i-1} + 6\mathbf{f}_i)/6 \\ \mathbf{a}_3 &= (-1\mathbf{f}_{i-3} + 3\mathbf{f}_{i-2} - 3\mathbf{f}_{i-1} + 1\mathbf{f}_i)/6 \end{aligned} \quad (5.15)$$

ein, so erhält man daraus die Adams-Bashforth Formel 4. Ordnung

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \int_{t_i}^{t_i+h} \mathbf{p}(t) dt = \mathbf{y}_i + \frac{h}{24}(-9\mathbf{f}_{i-3} + 37\mathbf{f}_{i-2} - 59\mathbf{f}_{i-1} + 55\mathbf{f}_i), \quad (5.16)$$

die auch bei größeren Schrittweiten eine gute Näherung des Zustandsvektors zur Zeit $t_{i+1} = t_i + h$ liefert. Durch wiederholte Anwendung erhält man wieder entsprechende Werte für nachfolgende Zeiten $t_i + jh$.

Verwendet man für das interpolierende Polynom mehr Stützstellen und damit einen höheren Grad, dann steigt damit im allgemeinen die Genauigkeit des Verfahrens. Für große Schrittweiten kann es dabei jedoch zu Instabilitäten kommen, so daß Ordnung und Schrittweite je nach gewünschter Genauigkeit passend gewählt werden müssen.

Das Verfahren DE, das im folgenden zur Berechnung von gestörten Kleinplanetbahnen eingesetzt wird, ist dem Prinzip nach ebenfalls ein Adams-Mehrschrittverfahren. Gegenüber der hier skizzierten einfachen Form eines Mehrschrittverfahrens fester Ordnung und Schrittweite weist DE jedoch einige Besonderheiten auf, die im praktischen Einsatz von großem Vorteil sind:

- Die Anzahl zurückliegender Funktionswerte, und damit die Ordnung des Verfahrens, ist nicht fest, sondern wird von DE selbstständig gewählt. Bei hohen Genauigkeitsforderungen arbeitet DE maximal mit zwölfter Ordnung.
- Die verwendete Schrittweite kann ebenfalls von einem Schritt zum nächsten variieren und so der zeitlichen Änderung des Zustandsvektors angepaßt werden. Dies ist besonders bei der Behandlung stark exzentrischer Bahnen von Nutzen, die in der Nähe des Perihels wesentlich kleinere Schrittweiten als im Aphel benötigen.
- Im Gegensatz zu einfachen Mehrschrittverfahren fester Ordnung und Schrittweite wird keine spezielle Anlaufrechnung benötigt, um die ersten Funktionswerte $\mathbf{f}_0, \mathbf{f}_1, \dots$ zu ermitteln. Stattdessen beginnt DE mit einem kleinen Schritt der Ordnung eins und erhöht dann mit jedem Schritt die Ordnung und die Schrittweite solange, bis die optimalen Werte erreicht sind. DE ist damit selbststartend.
- Der Zustandsvektor $\mathbf{y}(t)$ kann an beliebig dicht aufeinander folgenden Zeitpunkten ausgegeben werden, wobei die Werte an den intern verwendeten Stützpunkten automatisch interpoliert werden. Die Wahl der Schrittweite wird damit nicht von der Wahl der Ausgabepunkte beeinflußt.

Das Verfahren DE wurde von L. F. Shampine und M. K. Gordon entwickelt und als FORTRAN Programm implementiert. Aus Platzgründen kann hier weder auf den

umfangreichen Programmcode noch auf Details der Implementierung eingegangen werden. Der Leser sei hierzu auf das Buch „Computer-Lösung gewöhnlicher Differentialgleichungen“ verwiesen, in dem die Grundlagen des Verfahrens ausführlich erläutert sind.

Für die vorliegende Anwendung wurde DE in C++ übersetzt und an die sprachlichen Möglichkeiten und Besonderheiten dieser Sprache angepaßt. Große Vorteile bietet dabei die Definition einer eigenen Klasse SolverDE, die den Benutzer von administrativen Aufgaben wie der Bereitstellung des notwendigen Arbeitsspeichers praktisch vollständig entlastet:

```
// Statuscodes des Integrators SolverDE
enum DE_STATE {
    DE_INIT          = 1,      // Startschritt
    DE_DONE          = 2,      // Erfolgreicher Integrationsschritt
    DE_ACCURACY_NOT_ACHIEVED = 3, // Zu hohe Genauigkeitsforderung
    DE_TOO_MANY_STEPS   = 4,      // Zu viele Schritte benoetigt
    DE_STIFF          = 5,      // Verdacht auf steife Differentialgleichung
    DE_INVALID_PARAMS = 6,      // Ungueltige Eingaben
};

// 
// SolverDE Integrator-Klasse
//
class SolverDE
{
public:
    // Konstruktor, Destruktor
    SolverDE (DEFunct F, int Neqn);
    ~SolverDE ();
    // Integration
    void Integ (           // Alle Parameter sind nach Aufruf aktualisiert:
        double     y[],       // Loesung
        double&   t,          // Wert der unabhaengigen Variablen
        double     tout,       // Wert der unabh. Variablen,
                           // bis zu dem integriert werden soll
        double&  relerr,     // Gewuenschte relative Genauigkeit der Loesung
        double&  abserr,     // Gewuenschte absolute Genauigkeit der Loesung
        DE_STATE& State,     // Statuscode
        bool    PermitTOUT = true // Auswertung des Integranden bei TOUT erlaubt?
    );
private:
    // Datenelemente
    DEFunct  f;
    int      neqn,ns,k,kold;
    double   *yy,*wt,*p,*yp,*ypout,**phi;
    double   alpha[13],beta[13],v[13],w[13],psi[13],sig[14],g[14];
    double   x,h,hold,told,delsgn;
    bool    OldPermit, phase1,start,nornd;
    // Interpolation
    void Intrp ( double x, const double y[], double xout,
                  double yout[], double ypout[] );
    // Elementarer Integrationsschritt
    void Step (double& x, double y[], double& eps, bool& crash);
};
```

Insgesamt umfaßt das Integrationsverfahren die drei Methoden `Integ`, `Step` und `Intrp`, wobei lediglich `Integ` vom Benutzer direkt aufgerufen wird. Die beiden übrigen Elementfunktionen werden innerhalb von `Integ` verwendet und dienen zur Berechnung eines einzelnen Integrationsschritts (`Step`) und zur Interpolation des Zustandsvektors an den Ausgabezeitpunkten (`Intrp`).

Als Besonderheit der C++ Implementierung ist zu beachten, daß alle Felder innerhalb der Klasse `SolverDE` analog zum ursprünglichen Fortran Code beginnend mit dem untersten Index 1 angesprochen werden. Da C++ keine Möglichkeit frei wählbarer Feldgrenzen vorsieht, sind Vektoren der (logischen) Dimension n als Felder der (physischen) Dimension $[n+1]$ deklariert, wobei das erste Feldelement (Index [0]) nicht belegt wird.

Der Konstruktor `SolverDE` erwartet als Parameter das zu integrierende Differentialgleichungssystem und dessen Dimension. Erstes Argument ist dabei eine Funktion der Form

```
// Prototyp des Integranden DEfunct: void f (double x, double y[], double yp[])
typedef void (*DFunct)(
    double x,      // unabhaengige Variable
    double y[],   // Vektor mit Komponenten der Funktion
    double yp[]  // Vektor mit Komponenten der Ableitungen nach x
);
```

die zu einer Differentialgleichung $d\mathbf{y}/dx = \mathbf{f}(\mathbf{x}, \mathbf{y})$ bei gegebenen Argumenten $\mathbf{x} = x$ und $\mathbf{y} = \mathbf{y}$ die Ableitung $\mathbf{yp} = d\mathbf{y}/dx$ berechnet. Der zweite Parameter (`Neqn`) im Aufruf des Konstruktors bezeichnet die Anzahl der Differentialgleichungen und ist identisch mit dem obersten Feldindex der einzelnen Vektoren. Eine entsprechende Funktion für die Bewegungsgleichung eines Kleinplaneten kann mit den zu Beginn abgeleiteten Ausdrücken für die Beschleunigung als Funktion von Ort und Zeit wie folgt geschrieben werden:

```
-----
// F: Bewegungsgleichung in SolverDE-kompatibler Form
// X          Unabhaengige Variable (Zeit als Modifiziertes Julianisches Datum)
// Y[]        Zustandsvektor (Position und Geschwindigkeit)
// dYdX[]     Ableitung des Zustandsvektors nach X
//             (Geschwindigkeit und Beschleunigung)
-----
void F (double X, double Y[], double dYdX[])
{
    Vec3D a, r;
    r = Vec3D(Y[1], Y[2], Y[3]);
    a = Accel(X, r);
    dYdX[0] = 0.0;
    dYdX[1] = Y[4]; dYdX[2] = Y[5]; dYdX[3] = Y[6]; // Geschwindigkeit
    dYdX[4] = a[x]; dYdX[5] = a[y]; dYdX[6] = a[z]; // Beschleunigung
}
```

Der Zustandsvektor und seine Ableitung sind hierbei Größen der Dimension $n_{eqn} = 6$. Wie oben erläutert bedingt dies aber eine Deklaration von `Y` und `dYdX` als Felder der Dimension `Neqn+1`. Die Feldkomponenten mit Index [0] haben keine

weitere Bedeutung, werden aber zur Vermeidung uninitialisierter Variablen mit Null belegt.

Hat man, wie im Beispiel

```
SolverDE Orbit(F,Neqn);
```

ein Objekt der Klasse SolverDE angelegt, so kann die zugehörige Differentialgleichung anschließend durch Aufruf der Methode Integ integriert werden:

```
Orbit.Integ(y,t,tout,relerr,abserr,State,PermitTOUT);
```

Die einzelnen Parameter bezeichnen den Zustandsvektor (*y*), die Zeit (*t*), die Ausgabezeit (*tout*), die relative und die absolute Genauigkeitsforderung (*relerr*, *abserr*) sowie die Statusvariable (*State*). Der letzte Parameter *PermitTOUT* ist optional, und gibt an, ob bei der internen Integration über den Zeitpunkt *t_{out}* hinaus integriert werden kann. Dies ist immer dann der Fall, wenn die zu integrierende Differentialgleichung keine Unstetigkeitsstellen oder Pole aufweist. Entsprechend ist der Parameter standardmäßig als *true* vorbesetzt.

Beim ersten Aufruf von Integ wird in *t* der Anfangszeitpunkt und in *y* der zugehörige Zustandsvektor übergeben. Ferner sind der gewünschte Ausgabezeitpunkt *tout* und die benötigte relative und absolute Genauigkeit *relerr* und *abserr* anzugeben. Die Statusvariable *State* wird auf DE_INIT gesetzt, um dem Integrator mitzuteilen, daß mit der Integration eines neuen Problems begonnen wird.

Im Normalfall enthält *y* anschließend den gesuchten Zustandsvektor zur Zeit *tout*. Gleichzeitig wird *t* der Wert *tout* zugewiesen, während die Statusvariable *State* auf DE_DONE (=erfolgreicher Schritt) gesetzt wird. In allen nachfolgenden Schritten wird lediglich *tout* vom Benutzer ein neuer Wert zugewiesen, während die übrigen Variablen (insbesondere *State*) unverändert bleiben. Lediglich zu Beginn eines neuen Problems oder bei Umkehrung der Integrationsrichtung muß *State* wieder auf DE_INIT zurückgesetzt werden.

Liefert Integ am Ende eines Aufrufs einen anderen Wert als DE_DONE für *State* zurück, so hat dies folgende Bedeutung:

DE_ACCURACY_NOT_ACHIEVED Der Punkt *tout* wurde nicht erreicht, weil die geforderte Toleranz *relerr* bzw. *abserr* zu klein war. Beide Werte wurden für die weitere Rechnung geeignet erhöht.

DE_TOO_MANY_STEPS Der Punkt *tout* wurde nicht erreicht, weil dazu intern mehr als *maxnum*=500 Schritte erforderlich waren.

DE_STIFF *tout* wurde nicht erreicht, weil die Differentialgleichung steif zu sein scheint. Bei himmelsmechanischen Problemen sollte dieser Fall nicht auftreten.

DE_INVALID_PARAMS Unerlaubte Eingabeparameter (z.B. *t=tout*).

In allen Fällen, außer bei *State*=DE_INVALID_PARAMS, kann die Integration direkt durch wiederholten Aufruf von Integ ohne Änderung der Parameter fortgesetzt werden. Die Unterbrechung dient im wesentlichen dazu, den Benutzer auf Probleme aufmerksam zu machen und beispielsweise Endlosintegrationen zu vermeiden.

5.4 Oskulierende Bahnelemente

Im Falle einer ungestörten Keplerbahn ist die Bewegung eines Himmelskörpers um die Sonne durch sechs Bahnelemente (Halbachse a , Exzentrizität e , Bahnneigung i , Länge des aufsteigenden Knotens Ω , Länge des Perihels ω und mittlere Anomalie M) festgelegt. Die Bahnelemente sind dabei konstante Größen, aus denen Ort und Geschwindigkeit zu beliebigen Zeitpunkten berechnet werden können (vgl. Kap. 4).

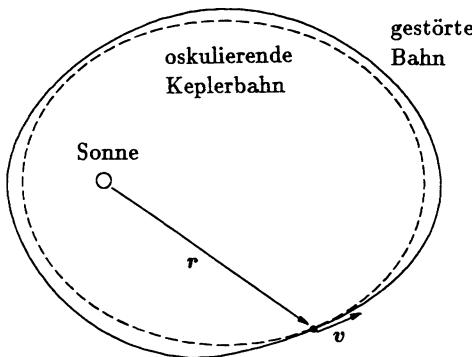


Abb. 5.3. Darstellung einer gestörten Bahn durch oskulierende Bahnelemente

Da die Bahnelemente aufgrund ihrer geometrischen Bedeutung besonders anschauliche Größen sind, liegt es nahe, sie auch zur Beschreibung einer gestörten Kleinplaneten- oder Kometenbahn zu verwenden. Hierbei macht man sich zunutze, daß Bahnelemente und Zustandsvektor in eineindeutiger Weise miteinander zusammenhängen. Gibt man zu einem bestimmten Zeitpunkt t den Ort $\mathbf{r} = (x, y, z)$ und die Geschwindigkeit $\dot{\mathbf{r}} = (\dot{x}, \dot{y}, \dot{z})$ eines Himmelskörpers vor, dann gibt es hierzu genau einen Satz von Bahnelementen $(a, e, i, \Omega, \omega, M)$, aus denen man nach den Gleichungen des ungestörten Keplerproblems den gegebenen Zustandsvektor $(\mathbf{r}, \dot{\mathbf{r}})$ erhält. Damit lassen sich auch für eine gestörte Bewegung Bahnelemente definieren. Diese sind allerdings nicht mehr konstant, sondern zeigen je nach Art und Stärke der Störung unterschiedliche zeitliche Schwankungen. Man bezeichnet die aus dem Zustandsvektor abgeleiteten Bahnelemente üblicherweise als *oskulierende* (anschmiegender) Bahnelemente, weil die zugehörige Keplerbahn zur jeweiligen Epoche die wahre Bahn berührt und für eine gewisse Zeitspanne in ihrer unmittelbaren Nähe verläuft (Abb. 5.3).

Die oskulierende Bahnebene ist – wie für die ungestörte Bewegung – zu jedem Zeitpunkt durch den Orts- und Geschwindigkeitsvektor festgelegt (siehe auch Abb. 4.1). Das Kreuzprodukt

$$\mathbf{h} = \mathbf{r} \times \dot{\mathbf{r}} = \begin{pmatrix} y\dot{z} - z\dot{y} \\ z\dot{x} - x\dot{z} \\ x\dot{y} - y\dot{x} \end{pmatrix} \quad (5.17)$$

von $\mathbf{r}(t)$ und $\dot{\mathbf{r}}(t)$ definiert deshalb einen Vektor $\mathbf{h}(t)$, der senkrecht auf der jeweiligen Bahnebene und damit parallel zum Gaußschen Bahnnormalenvektor

$$\mathbf{R} = \begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix} = \begin{pmatrix} +\sin i \sin \Omega \\ -\sin i \cos \Omega \\ +\cos i \end{pmatrix} = \begin{pmatrix} h_x/h \\ h_y/h \\ h_z/h \end{pmatrix} \quad (5.18)$$

verläuft. Hieraus ergeben sich die oskulierende Bahnneigung i und die Länge des aufsteigenden Knotens Ω zu

$$i = \tan^{-1} \left(\frac{\sqrt{h_x^2 + h_y^2}}{h_z} \right) \quad \Omega = \tan^{-1} \left(\frac{h_x}{-h_y} \right) \quad . \quad (5.19)$$

Durch Auflösen von (4.20) nach u ergibt sich zusätzlich die Beziehung

$$u = \nu + \omega = \tan^{-1} \left(\frac{zh}{-xh_y + yh_x} \right) \quad (5.20)$$

für das Argument der Breite u , die später zur Berechnung des Arguments des Perihels benötigt wird.

Die große Halbachse der Bahn hängt, wie man durch Einsetzen der entsprechenden Ausdrücke (4.5) und (4.10) bestätigt, nur von der Entfernung r von der Sonne und dem Betrag $v = |\dot{\mathbf{r}}|$ der Geschwindigkeit ab:

$$a = \left(\frac{2}{r} - \frac{v^2}{GM_{\odot}} \right)^{-1} \quad . \quad (5.21)$$

Eine Ellipse mit positiver Halbachse als oskulierende Keplerbahn ergibt sich demnach immer dann, wenn die Geschwindigkeit des Körpers relativ zur Sonne kleiner als die jeweilige Fluchtgeschwindigkeit $v_{\infty} = \sqrt{2GM_{\odot}/r}$ ist.

In diesem Fall erhält man durch Umformung der beiden Gleichungen (4.5) und (4.10) für Ort und Geschwindigkeit als Funktion der exzentrischen Anomalie die zwei Beziehungen

$$\begin{aligned} e \cos(E) &= 1 - r/a \\ e \sin(E) &= (x\dot{x} + y\dot{y} + z\dot{z})/\sqrt{GM_{\odot}a} \end{aligned} \quad (5.22)$$

zur Bestimmung der Exzentrizität e und der exzentrischen Anomalie E . Mit Hilfe der Keplergleichung (4.7) folgt daraus dann die gesuchte mittlere Anomalie

$$M = E - \frac{180^\circ}{\pi} e \sin(E) \quad (5.23)$$

zur Zeit t . Schließlich erhält man über die zu E gehörige wahre Anomalie

$$\nu = \tan^{-1} \frac{\sqrt{1 - e^2} \sin(E)}{\cos(E) - e} \quad (5.24)$$

als sechstes Bahnelement das Argument des Perihels

$$\omega = \nu - u \quad . \quad (5.25)$$

Für parabolische und hyperbolische Bahnen gelten entsprechende Beziehungen, die jedoch für die Behandlung von Kleinplanetenbahnen oder Bahnen periodischer Kometen nicht weiter benötigt werden. Die folgende Funktion `Elements` beschränkt sich deshalb lediglich auf elliptische Bahnen.

```

//-----
// Elements: Berechnet die Elemente einer elliptischen Bahn aus Orts- und
// Geschwindigkeitsvektor
// GM      Produkt aus Gravitationskonstante und Zentralmasse [AE^3*d^-2]
// r       Heliozentrischer ekliptikaler Ort in [AE]
// v       Heliozentrische ekliptikale Geschwindigkeit in [AE/d]
// a       Grosse Halbachse der Bahn in [AE]
// e       Exzentrizitaet der Bahn
// i       Neigung der Bahn zur Ekliptik in [rad]
// Omega   Laenge des aufsteigenden Knotens der Bahn in [rad]
// omega   Argument des Perihels in [rad]
// M       Mittlere Anomalie in [rad]
//-----

void Elements ( double GM, const Vec3D& r, const Vec3D& v,
                double& a, double& e, double& i,
                double& Omega, double& omega, double& M )
{
    // Variablen

    Vec3D h;
    double H, u, R, v2;
    double eCosE, eSinE, e2, E, nu;

    h = Cross(r,v);                                // Flaechengeschwindigkeit
    H = Norm(h);
    Omega = atan2(h[x], -h[y]);                   // Laenge aufst. Knoten
    i     = atan2(sqrt(h[x]*h[x]+h[y]*h[y]), h[z]); // Bahnneigung
    u     = atan2(r[z]*H, -r[x]*h[y]+r[y]*h[x]);  // Argument der Breite

    R   = Norm(r);                                // Entfernung
    v2 = Dot(v, v);                               // Quadrat der Geschw.
    a  = 1.0 / (2.0/R-v2/GM);                     // Grosse Halbachse

    eCosE = 1.0-R/a;                            // e*cos(E)
    eSinE = Dot(r, v)/sqrt(GM*a);               // e*sin(E)
    e2   = eCosE*eCosE +eSinE*eSinE;            // Exzentrizitaet
    E    = atan2(eSinE,eCosE);                  // Exzentrische Anomalie

    M   = E - eSinE;                            // Mittlere Anomalie
    nu = atan2(sqrt(1.0-e2)*eSinE, eCosE-e2); // Wahre Anomalie
    omega = u - nu;                            // Argument des Perihels

    if (Omega<0.0) Omega += 2.0*pi;
    if (omega<0.0) omega += 2.0*pi;
    if (M <0.0) M += 2.0*pi;

};
```

5.5 Das Programm NUMINT

Das Programm Numint dient zur genauen Berechnung von Planetoidenbahnen oder periodischen Kometenbahnen und lehnt sich im Aufbau und in der Bedienung an das in Kap. 4 behandelte Programm Comet an. Durch die Verwendung eines numerischen Integrationsverfahrens können neben der Anziehungskraft der Sonne auch die Störungen durch die großen Planeten in der Bahnvorhersage berücksichtigt werden. Besonders bei längeren Vorhersagezeiträumen lassen sich so deutlich genauere Ergebnisse erzielen. Der höheren Genauigkeit steht allerdings der Nachteil gegenüber, daß die zur Integration benötigte Rechenzeit proportional zur Länge der Ephemeride anwächst und im allgemeinen deutlich höher ist als die des Programms Comet.

Beim Aufruf des Programms werden zunächst die Bahnelemente und das zugehörige Äquinoktium aus der Datei `Numint.dat` gelesen und zur Kontrolle auf dem Schirm ausgegeben, bevor der Benutzer die Zeitspanne, die Schrittweite und das gewünschte Äquinoktium der Ephemeride eingibt. Aus diesen Angaben werden dann die Position und die Geschwindigkeit des Kleinplaneten bezogen auf das feste Äquinoktium J2000 bestimmt und durch numerische Integration von der Epoche der Bahnelemente bis zum gewünschten Beginn der Ephemeride vorhergesagt. Arbeitet man mit Bahnelementen, deren Epoche einige Jahre vor dem Startdatum liegt, dann kann die Integration je nach Leistungsfähigkeit des Rechners durchaus einige Minuten in Anspruch nehmen.

Aus dem so erhaltenen Zustandsvektor zu Beginn der Ephemeride berechnet Numint dann zunächst die zugehörigen oskulierenden Bahnelemente und gibt diese auf dem Bildschirm aus. Aus den Änderungen der einzelnen Elemente gegenüber den Werten zur ursprünglichen Epoche erhält man so einen guten Eindruck von der Wirkung der Störungen, die von den großen Planeten ausgeübt werden. Dariüberhinaus kann man die so erhaltenen Bahnelemente nutzen, um bei wiederholten Rechnungen mit alten Bahnelementen nicht jedesmal große Zeitspannen bis zum Beginn der Ephemeride überbrücken zu müssen.

Anschließend wird analog zum Programm Comet eine Ephemeride des Kleinplaneten mit der gewünschten Schrittweite und Zeitdauer berechnet und ausgegeben. Die Lichtlaufzeit wird dabei berücksichtigt, so daß die resultierenden, astronomischen Koordinaten direkt mit Sternkarten oder Sternkatalogen verglichen werden können. Die Position der Erde relativ zur Sonne wird – wie in Comet – mit Hilfe der Funktion `SunPos` berechnet, um Fehler bei der Umrechnung von heliozentrischen in geozentrische Koordinaten des Kleinplaneten zu vermeiden.

```
-----  
// Datei: Numint.cpp  
// Zweck: Numerische Integration gestoerter Kleinplanetenbahnen  
// (c) 1999 Oliver Montenbruck, Thomas Pfleger  
-----  
  
#include <cmath>  
#include <fstream>  
#include <iomanip>  
#include <iostream>
```

```

#include "APC_Const.h"
#include "APC_DE.h"
#include "APC_IO.h"
#include "APC_Kepler.h"
#include "APC_Math.h"
#include "APC_Planets.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"

using namespace std;

-----
// WriteElm: Ausgabe der Bahnelemente
//   MjdEpoch Epoche als Modifiziertes Julianisches Datum
//   a          Grosse Halbachse der Bahn in [AE]
//   e          Exzentrizitaet der Bahn
//   i          Bahneigung gegen die Ekliptik in [rad]
//   Omega      Laenge des aufsteigenden Knotens der Bahn in [rad]
//   omega      Argument des Perihels in [rad]
//   M          Mittlere Anomalie in [rad]
//   T_eqx0    Aequinoktium in Julianischen Jahrhunderten seit J2000
//-----
void WriteElm ( double MjdEpoch, double a, double e, double i,
                double Omega, double omega, double M, double T_eqx0 )
{
    cout << "  Epoche (J M T)           "
        << "  " << setprecision(2) << DateTime(MjdEpoch,DDd) << endl;
    cout << "  Grosse Halbachse (a)       "
        << setprecision(7) << setw(14) << a << " AE" << endl;
    cout << "  Exzentrizitaet (e)         "
        << setprecision(7) << setw(14) << e << endl;
    cout << "  Bahneigung (i)            "
        << setprecision(5) << setw(12) << i*Deg << " Grad" << endl;
    cout << "  Laenge des aufst. Knotens "
        << setprecision(5) << setw(12) << Omega*Deg << " Grad" << endl;
    cout << "  Argument des Perihels   "
        << setprecision(5) << setw(12) << omega*Deg << " Grad" << endl;
    cout << "  Mittlere Anomalie (M)    "
        << setprecision(5) << setw(12) << M*Deg      << " Grad" << endl;
    cout << "  Aequinoktium              "
        << setprecision(2) << setw(9)  << 2000.0+100*T_eqx0 << endl;
    cout << endl;
}

-----
// GetElm: Einlesen der Bahnelemente aus einer Datei
//   Filename  Name der Eingabedatei mit den Bahnelementen
//   MjdEpoch  Epoche als Modifiziertes Julianisches Datum
//   a          Grosse Halbachse der Bahn in [AE]
//   e          Exzentrizitaet der Bahn
//   M          Mittlere Anomalie in [rad]

```

```

// PQR      Transformationsmatrix Bahnebene -> Ekliptik
// T_eqx0   Aequinoktium der Bahnelemente
//-----
void GetElm ( char* Filename,
              double& MjdEpoch,
              double& a, double& e, double& M, Mat3D& PQR, double& T_eqx0 )
{
    int      year, month;
    double   Omega, i, omega;
    double   Year, day;
    ifstream inp;
    // Bahnelemente lesen
    inp.open(Filename);
    inp >> year >> month >> day; inp.ignore(81,'\'n');
    inp >> a; inp.ignore(81,'\'n');
    inp >> e; inp.ignore(81,'\'n');
    inp >> i; inp.ignore(81,'\'n');
    inp >> Omega; inp.ignore(81,'\'n');
    inp >> omega; inp.ignore(81,'\'n');
    inp >> M; inp.ignore(81,'\'n');
    inp >> Year; inp.ignore(81,'\'n');
    inp.close();
    // Abgeleitete Daten berechnen
    i*=Rad;
    Omega*=Rad;
    omega*=Rad;
    M*=Rad;
    MjdEpoch = Mjd(year,month,int(day))+(day-int(day));
    T_eqx0   = (Year-2000.0)/100.0;
    PQR      = GaussVec(Omega,i,omega);
    // Bahnelemente anzeigen
    cout << " Bahnelemente aus der Datei " << Filename << ":" << endl << endl;
    WriteElm ( MjdEpoch, a,e,i,Omega,omega,M, T_eqx0 );
}

//-----
// GetEph: Eingabe des Zeitraums der Ephemeride, der Schrittweite und des
//          Aequinoktiums
// MjdStart  Anfangsdatum der Ephemeride als MJD
// Step       Schrittweite der Ephemeride in [d]
// MjdEnd    Enddatum der Ephemeride als MJD
// T_eqx     Gewuenschtes Aequinoktium der Ephemeride in Julianischen
//          Jahrhunderten seit J2000
//-----
void GetEph (double& MjdStart, double& Step, double& MjdEnd, double& T_eqx)
{
    int      year, month, day;
    double hour, Year;
    cout << " Beginn und Ende der Ephemeride: " << endl;
    cout << " Erstes Berechnungsdatum (JJJJ MM TT HH.HHH)      ... ";
    cin >> year >> month >> day >> hour; cin.ignore(81,'\'n');
    MjdStart = Mjd(year,month,day) + hour/24.0 ;
    cout << " Letztes Berechnungsdatum (JJJJ MM TT HH.HHH)      ... ";
    cin >> year >> month >> day >> hour; cin.ignore(81,'\'n');
    MjdEnd = Mjd(year,month,day) + hour/24.0 ;
}

```

```

cout << " Schrittweite (TT HH.HH) ... ";
cin >> day >> hour; cin.ignore(81,'\'n');
Step = day + hour/24.0;
cout << endl
    << " Gewuenschtes Aequinoktium der Ephemeride (JJJJ.J) ... ";
cin >> Year; cin.ignore(81,'\'n');
T_eqx = (Year-2000.0) /100.0;
}

//-----
// Hauptprogramm
//
//-----

void main(int argc, char* argv[])
{
    // Konstanten
    const pol_index R      = pol_index(2);    // Index fuer Vektorlaenge
    const int      Neqn = 6;                   // Anzahl der Gleichungen
    const double   eps   = 1.0e-10;            // Relative Genauigkeit

    // Variablen
    int          n_line;
    double       MjdStart, Step, MjdEnd, T_eqx;
    double       MjdEpoch, a, e, M, T_eqx0;
    double       Omega, i, omega;
    Mat3D       PQR, P;
    double       Mjd, MjdStep, T;
    SolverDE   Orbit(F, Neqn);
    double      Y[Neqn+1];
    double      relerr,abserr;
    DE_STATE   State;
    Vec3D       r, v;
    Vec3D       R_Sun, r_helioc, v_helioc, r_geoc, r_equ;
    double      dist, fac;
    char        InputFile[APC_MaxFilename] = "";
    char        OutputFile[APC_MaxFilename] = "";
    bool        FoundInputfile = false;
    bool        FoundOutputfile = false;
    ofstream    OutFile;

    // Titel
    cout << endl
        << " NUMINT: Numerische Integration gestoerter Kleinplanetenbahnen"
        << endl
        << "           (c) 1999 Oliver Montenbruck, Thomas Pfleger "
        << endl << endl;

    // Ermittle die Namen der Ein- und Ausgabedatei
    GetFilenames( argc, argv, "Numint.dat", InputFile, FoundInputfile,
                  OutputFile, FoundOutputfile );
    // Abbruch, falls die Eingabedatei nicht gefunden wurde
    if (!FoundInputfile) {
        cerr << " Abbruch. Eingabedatei nicht gefunden." << endl; exit(-1);
    }
}

```

```

// Bahnelemente lesen, Zeitraum und Aequinoktium eingeben
GetElm ( InputFile, MjdEpoch, a, e, M, PQR, T_eqx0 );
GetEph ( MjdStart, Step, MjdEnd, T_eqx );
// Bei Bedarf die Ausgabe in eine Datei umlenken
if (FoundOutputfile) {
    OutFile.open(OutputFile); if (OutFile.is_open()) cout = OutFile;
}

P = PrecMatrix_Ecl(T_J2000,T_eqx);

// Anfaenglicher Zustandsvektor (Ekliptik und Aequinoktium J2000)
PQR = PrecMatrix_Ecl(T_eqx0,T_J2000) * PQR;
Ellip ( GM_Sun, M,a,e, r,v );
r = PQR*r;
v = PQR*v;
Y[0]=0.0; // Nicht verwendet
Y[1]=r[x]; Y[2]=r[y]; Y[3]=r[z];
Y[4]=v[x]; Y[5]=v[y]; Y[6]=v[z];
Mjd = MjdEpoch;

// Start der Integration: Zustandsvektor von der Epoche
// bis zum Beginn der Ephemeride propagieren
State = DE_INIT;
relerr = eps; abserr = 0.0;
do {
    Orbit.Integ(Y, Mjd, MjdStart, relerr, abserr, State);
    if ( State==DE_INVALID_PARAMS ) {
        cerr << "Abbruch. Ungueltige Parameter." << endl;
        exit(1);
    }
}
while ( State > DE_DONE );

// Bahnelemente zu Beginn der Ephemeride (Aequinoktium T_eqx)
r = P*Vec3D(Y[1],Y[2],Y[3]);
v = P*Vec3D(Y[4],Y[5],Y[6]);
Elements (GM_Sun, r, v, a, e, i, Omega, omega, M);

// Ausgabe der Bahnelemente
cout << endl << endl
    << " Bahnelemente zur Anfangsepoke:" << endl
    << endl;
WriteElm ( Mjd, a,e,i,Omega,omega,M, T_eqx );

// Vorspann der Ephemeride
cout << endl << endl
    << "     Datum      ET   Sonne      l      b      r"
    << "           RA          Dec      Entfernung " << endl
    << setw(43) << " " << "     h   m   s       o   ' \n"      (AE) " << endl;

// Ephemeride berechnen
n_line = 0;
MjdStep = MjdStart;

```

```

// Schleife ueber Berechnungszeitpunkte
while ( MjdStep < MjdEnd + Step/2 ) {

    // Integriere die Bahn bis MjdStep
    do {
        Orbit.Integ(Y, Mjd, MjdStep, relerr, abserr, State);
        if (State==DE_INVALID_PARAMS) {
            cerr << "Abbruch. Ungueltige Parameter." << endl;
            exit(1);
        }
    }
    while ( State > DE_DONE );

    // Heliozentrische ekliptikale Koordinaten, Aequinoktium T_eqx
    r_helioc = P*Vec3D(Y[1],Y[2],Y[3]);
    v_helioc = P*Vec3D(Y[4],Y[5],Y[6]);

    // Geozentrische ekliptikale Koordinaten der Sonne, Aequinoktium T_eqx
    T = ( Mjd - MJD_J2000 ) / 36525.0;
    R_Sun = PrecMatrix_Ecl(T,T_eqx) * SunPos(T);

    // Geometrische geozentrische Koordinaten
    r_geoc = r_helioc + R_Sun;

    // Korrektur (1. Ordnung) fuer die Lichtlaufzeit
    dist = Norm(r_geoc);
    fac = 0.00578*dist;
    r_geoc = r_geoc - fac*v_helioc;

    // Aequatoriale Koordinaten
    r_equ = Ecl2EquMatrix(T_eqx) * r_geoc;

    // Ausgabe
    cout << DateTime(Mjd,HHh)
        << fixed << setprecision(1)
        << setw(7) << Deg*R_Sun[phi]
        << setw(7) << Deg*r_helioc[phi]
        << setw(6) << Deg*r_helioc[theta]
        << setprecision(3) << setw(7) << r_helioc[R]
        << setprecision(1) << setw(12) << Angle(Deg*r_equ[phi]/15.0,DMMSSs)
        << " " << showpos << setw(9) << Angle(Deg*r_equ[theta],DMMSS)
        << noshowpos << setprecision(6) << setw(11) << dist
        << endl;
    ++n_line;
    if ( (n_line % 5) ==0 ) cout << endl; // Leerzeile alle 5 Zeilen

    MjdStep += Step; // Naechster Berechnungszeitpunkt
}

if (OutFile.is_open()) OutFile.close();
}

```

Ausgehend von oskulierenden Bahnelementen aus dem Jahr 1983 soll nun als Beispiel für die Anwendung des Programms eine Ephemeride des Kleinplaneten Ceres für den Oppositionszeitraum des Jahres 1992 berechnet werden (vgl.

Tabelle 5.3. Oskulierende Bahnelemente des Kleinplaneten Ceres

Epoche	1983 Sep. 23.0 ET	1992 Jun. 27.0 ET
a	2.7657991 AE	2.7674385 AE
e	0.0785650	0.0765551
i (1950)	10°60646	10°59950
(2000)	10°60695	10°59999
Ω (1950)	80°05225	80°01288
(2000)	80°71588	80°67649
ω (1950)	73°07274	71°07929
(2000)	73°10812	71°11469
M	174°19016	141°46349

Tabelle 5.3). Die Eingabedatei `Numint.dat` enthält in der ersten Zeile die Epoche der Bahnelemente, anschließend die sechs Bahnelemente in der Reihenfolge $a, e, i, \Omega, \omega, M$ und zuletzt das Äquinoktium der Elemente:

```
1983 09 23.0 ! Epoche (Jahr Monat Tag.Tagesbruchteil) Kleinplanet 1 Ceres
  2.7657991 ! Grosse Halbachse a in AE
  0.0785650 ! Exzentrizitaet e
  10.60646 ! Bahnneigung i (Grad)
  80.05225 ! Laenge des aufsteigenden Knotens (Grad)
  73.07274 ! Argument des Perihels (Grad)
  174.19016 ! Mittlere Anomalie (Grad)
  1950.0   ! Aequinoktium der Bahnelemente
```

Entsprechende Angaben für andere numerierte Kleinplaneten werden jährlich in den „Ephemerides of Minor Planets“ (EMP) des Instituts für theoretische Astronomie in St. Petersburg (ehem. Leningrad), den „Minor Planet Circulars“ (MPC) des Smithsonian Astrophysical Observatory sowie in vielen anderen Jahrbüchern veröffentlicht.

Die Eingabe der gewünschten Zeitspanne und Schrittweite der Ephemeride sowie des Äquinoktiums verläuft wie beim Programm `Comet`. Im folgenden Ausdruck sind die Benutzereingaben wie gewohnt durch kursive Schrift hervorgehoben.

```
NUMINT: Numerische Integration gestoerter Kleinplanetenbahnen
(c) 1999 Oliver Montenbruck, Thomas Pfleger
```

Standard-Eingabedatei `Numint.dat`

Epoche (J M T)	1983/09/23.00
Grosse Halbachse (a)	2.7657991 AE
Exzentrizitaet (e)	0.0785650
Bahnneigung (i)	10.60646 Grad
Laenge des aufst. Knotens	80.05225 Grad
Argument des Perihels	73.07274 Grad
Mittlere Anomalie (M)	174.19016 Grad
Aequinoktium	1950.00

Beginn und Ende der Ephemeride:

Erstes Berechnungsdatum (JJJJ MM TT HH.HHH)	... 1992 06 27 00.0
Letztes Berechnungsdatum (JJJJ MM TT HH.HHH)	... 1992 07 25 00.0
Schrittweite (TT HH.HH)	... 2 00.0

Gewuenschtes Aequinoktium der Ephemeride (JJJJ.J) ... 2000.0

Bahnelemente zur Anfangsepoke:

Epoche (J M T)	1992/06/27 00.00
Grosse Halbachse (a)	2.7674372 AE
Exzentrizitaet (e)	0.0765610
Bahnneigung (i)	10.59999 Grad
Laenge des aufst. Knotens	80.67653 Grad
Argument des Perihels	71.11641 Grad
Mittlere Anomalie (M)	141.46191 Grad
Aequinoktium	2000.00

Datum	ET	Sonne	l	b	r	RA			Dec	Entfernung
						h	m	s		
1992/06/27 00.0	95.7	297.9	-6.5	2.939	20 55 54.5	-27	00	46	2.042458	
1992/06/29 00.0	97.6	298.2	-6.5	2.940	20 54 53.4	-27	13	47	2.029192	
1992/07/01 00.0	99.5	298.6	-6.6	2.941	20 53 46.7	-27	26	56	2.016828	
1992/07/03 00.0	101.4	299.0	-6.6	2.942	20 52 34.7	-27	40	13	2.005397	
1992/07/05 00.0	103.3	299.4	-6.7	2.942	20 51 17.5	-27	53	32	1.994924	
1992/07/07 00.0	105.2	299.7	-6.7	2.943	20 49 55.6	-28	06	53	1.985432	
1992/07/09 00.0	107.1	300.1	-6.8	2.944	20 48 29.0	-28	20	11	1.976941	
1992/07/11 00.0	109.0	300.5	-6.8	2.945	20 46 58.3	-28	33	24	1.969468	
1992/07/13 00.0	111.0	300.9	-6.9	2.946	20 45 23.6	-28	46	28	1.963030	
1992/07/15 00.0	112.9	301.2	-6.9	2.946	20 43 45.4	-28	59	22	1.957641	
1992/07/17 00.0	114.8	301.6	-7.0	2.947	20 42 03.9	-29	12	03	1.953316	
1992/07/19 00.0	116.7	302.0	-7.0	2.948	20 40 19.7	-29	24	26	1.950066	
1992/07/21 00.0	118.6	302.4	-7.1	2.949	20 38 33.1	-29	36	31	1.947903	
1992/07/23 00.0	120.5	302.7	-7.1	2.950	20 36 44.6	-29	48	14	1.946836	
1992/07/25 00.0	122.4	303.1	-7.2	2.950	20 34 54.5	-29	59	32	1.946871	

Das obige Beispiel geht davon aus, daß bei der Berechnung der Beschleunigungen in Accel die genauen Planetenpositionen aus PertPosition verwendet werden. Nach Beendigung der Eingabe ist für die Integration der Bahn über neun Jahre deshalb je nach Rechner eine gewisse Zeit erforderlich, bevor die oskulierenden Bahnelemente für den 27.6.1992 ausgegeben werden. Die anschließende Berechnung der einmonatigen Ephemeride benötigt dagegen nur Bruchteile einer Sekunde.

Wie ein Vergleich mit dem Jahrbuch zeigt, beträgt die erreichte Genauigkeit der Ephemeride rund 1" und liegt damit in derselben Größenordnung wie die Genauigkeit der Sonnenkoordinaten, die bei der Umrechnung von heliozentrischen zu geozentrischen Koordinaten verwendet werden.

Rechnet man ohne die Störungen der Planeten, indem man zum Beispiel die entsprechenden Zeilen in der Funktion Accel auskommentiert, dann erhält man zum Vergleich die folgende Ephemeride:

Datum	ET	Sonne	l	b	r	RA	Dec	Entfernung						
								h	m	s	o	'	"	(AE)
1992/06/27	00.0	95.7	298.6	-6.6	2.938	21 00 35.9	-26 49 55	2.049522						
1992/06/29	00.0	97.6	299.0	-6.6	2.939	20 59 37.9	-27 02 56	2.035891						
...														
1992/07/23	00.0	120.5	303.5	-7.2	2.950	20 41 55.4	-29 39 16	1.948277						
1992/07/25	00.0	122.4	303.9	-7.3	2.951	20 40 06.3	-29 50 51	1.947831						

Man erkennt, daß sich die Störungen mit $0^{\circ}7$ – $0^{\circ}8$ im wesentlichen in der heliozentrischen Länge des Kleinplaneten zeigen, während die Breite mit rund $0^{\circ}1$ geringere, aber immer noch merkliche Unterschiede aufweist. Die Rektaszension der ungestörten Ephemeride weicht aufgrund der geringen Entfernung zur Erde sogar um rund fünf Minuten oder $1^{\circ}3$ Grad von der tatsächlichen Bahn ab.

5.6 Die Asteroid Orbital Elements Datenbank

Bis heute wurden bereits über 50 000 Kleinplaneten entdeckt, deren Bahnen regelmäßig durch neue Beobachtungen verbessert werden. Verantwortlich für ihre Registrierung und Katalogisierung ist das Minor Planet Center am Smithsonian Astrophysical Observatory in Massachusetts. Daneben stellt auch das Lowell Observatory einen umfangreichen, frei zugänglichen Katalog mit Bahnelementen und weiteren Informationen bereit, der von Ted Bowell betreut und ständig aktualisiert wird. Mit Einverständnis der Herausgeber steht dieser Katalog einschließlich einer ausführlichen Beschreibung auch auf der beiliegenden CD-ROM zur Verfügung¹. Die Datei `astorb.dat` umfaßt in der vorliegenden Version (Stand Mai 1999) knapp 50 000 Datensätze mit einer Zeilenlänge von 267 Zeichen und kann mit einem entsprechenden Texteditor (Dateigröße ca. 13 MB) direkt gelesen und bearbeitet werden. Die Bedeutung der wichtigsten Angaben ist in Tabelle 5.4 zusammengefaßt.

Tabelle 5.4. Struktur der Asteroid Orbital Elements Datenbank (Auszug)

Spalten	Beschreibung
1– 5	Nummer des Kleinplaneten
7– 24	Name oder Bezeichnung
42– 46	Helligkeit [mag]
106–113	Epoche der Bahnelemente (jjjjmmtt)
115–124	Mittlere Anomalie [°]
126–135	Argument der Breite (J2000) [°]
137–146	Länge des aufsteigenden Knotens (J2000) [°]
148–156	Bahnneigung (J2000) [°]
158–167	Exzentrizität [°]
169–180	Halbachse [AE]

¹Aktuelle Versionen der AOE Datenbank sind bei Bedarf im Internet unter der Adresse <ftp://ftp.lowell.edu/pub/elgb/astorb.dat> oder .../astorb.gz erhältlich.

Zum einfacheren Gebrauch der Datenbank können einzelne Asteroiden mit Hilfe des Programms AOEcator nach Nummer oder Bezeichnung ausgewählt werden:

AOEcator: Auswahl von Kleinplaneten aus der Asteroid Orbital Elements Database
(c) 1999 Oliver Montenbruck, Thomas Pfleger

Name/Nummer des Kleinplaneten: *Lutetia*

1999 05 02.0	! Epoche	Kleinplanet (21) Lutetia
2.43629828	! a [AE]	
0.16181942	! e	
3.065874	! i [Grad]	
80.948968	! Omega [Grad]	
250.045037	! omega [Grad]	
251.120884	! M [Grad]	
2000.0	! Aequinoktium	

Das Ausgabeformat ist so gewählt, daß die Bahnelemente direkt vom Programm Numint weiter verarbeitet werden können. Dazu kann beim Start von AOEcator auch der Name einer Ausgabedatei angegeben werden (vgl. Abschn. A.1.3). Bei der Eingabe von Kleinplanetennummern ist zu beachten, daß vor und nach einer Nummer keine Leerzeichen stehen dürfen. Bei Namen (z.B. *Tycho Brahe*) und anderen Katalogbezeichnungen (z.B. 5073 T-3) sind Leerzeichen dagegen verbindlich. Buchstaben werden in Groß- und Kleinschreibweise akzeptiert. Der vollständige Quellcode von AOEcator ist zusammen mit dem der übrigen Programme auf der CD zum Buch enthalten.

6. Planetenbahnen

Sieht man einmal von Merkur und Pluto ab, dann besteht eine wesentliche Gemeinsamkeit der großen Planeten in den geringen Exzentrizitäten ihrer Bahnen und den kleinen Bahnneigungen gegenüber der Ekliptik. Im Gegensatz dazu sind die Bahnen der Kometen regellos im Raum verteilt und von sehr unterschiedlicher Form. Die kreisähnlichen Bahnen der Planeten sind darüber hinaus soweit getrennt, daß sie sich gegenseitig nie besonders nahe kommen können.

Diese Eigenschaften der Planetenbahnen haben zwei wichtige Konsequenzen für die Ephemeridenrechnung. Die erste betrifft die Behandlung der ungestörten Keplerbahn, wie sie in Kap. 4 für allgemeine Bahnformen beschrieben wurde. Aufgrund der niedrigen Exzentrizität zeigt die wahre Anomalie nur geringe periodische Abweichungen von der mittleren Anomalie. Sie kann deshalb schneller über eine Reihenentwicklung als durch Lösung der Keplergleichung bestimmt werden. Gleches gilt für die Entfernung von der Sonne, die sich im allgemeinen nur im Prozentbereich vom Wert der großen Halbachse unterscheidet. Weiterhin läßt sich die Abweichung der Bahn von der Ekliptik in einer kurzen Näherung darstellen, so daß man auf die aufwendige Berechnung der Gaußschen Vektoren verzichten kann. Diese Näherungen werden im nächsten Abschnitt etwas ausführlicher behandelt. Der zweite Punkt betrifft die Änderung der Planetenbahnen durch die gegenseitigen Gravitationskräfte. Da nahe Begegnungen der Planeten nicht vorkommen können, sind diese Kräfte immer um mehrere Größenordnungen kleiner als die Anziehung der Sonne. Wirklich starke Bahnänderungen, wie man sie von Kometen nach Vorübergängen an Jupiter oder Saturn kennt, treten in der Bewegung der Planeten nicht auf. Ihre Bahnen lassen sich deshalb durch mittlere Keplerellipsen beschreiben, denen kleine periodische Störungen überlagert sind.

Im allgemeinen werden Planetenephemeriden heute mit Hilfe numerischer Integrationsverfahren berechnet und anschließend auf Magnetbändern in geeigneter Weise abgespeichert. Dank des Einsatzes moderner Rechner zeichnet sich diese Methode durch große Einfachheit und Genauigkeit aus. Dem steht jedoch der Nachteil gegenüber, daß man keine analytische Darstellung der Planetenbewegung mehr zur Verfügung hat, wie dies beim reinen Zweikörperproblem der Fall ist. Eine der großen Leistungen der Himmelsmechanik war und ist deshalb die Aufstellung von Reihenentwicklungen, aus denen sich die Abweichungen der Planetenbahnen von der ungestörten Bewegung zu jeder Zeit berechnen lassen. Wie man schon aus der Länge dieser Reihen erahnen kann, ist ihre Herleitung eine ausgesprochen mühselige und aufwendige Arbeit, die weit über den Rahmen dieses Buches hinausgehen würde. Für unsere Zwecke – die genaue Berechnung von Planetenpositionen – genügt aber ein Verständnis für die grundlegende Struktur der

Reihenentwicklungen und ihre rechnerische Auswertung. Wer sich darüber hinaus für die Herleitung der Störungsreihen interessiert, muß auf die Lehrbücher der Himmelsmechanik verwiesen werden.

Zu den bekanntesten analytischen Darstellungen der Planetenbewegung gehören die Tafeln zur Bewegung der inneren Planeten Merkur bis Mars von Simon Newcomb vom US Naval Observatory. Obwohl sie bereits um die Jahrhundertwende aufgestellt wurden, bildeten sie noch bis vor wenigen Jahren die Grundlage für die entsprechenden Ephemeriden des *Astronomical Almanac*. Ähnliche Tafeln der äußeren Planeten gibt es von G.W. Hill für Jupiter und Saturn sowie von S. Newcomb für Uranus und Neptun. Sie wurden allerdings schon früh durch die ersten numerisch integrierten Ephemeriden abgelöst, da sie über längere Zeiträume nicht dieselbe Genauigkeit erreichten, wie die Tafeln der inneren Planeten.

Hinweis: Die genannten Werke bilden zusammen mit der Reihendarstellung der Plutobahn von E.Goffin, J.Meeus und C.Steyart die Grundlage dieses Kapitels. Die jeweiligen Formulierungen der Störsterme unterscheiden sich für die einzelnen Planeten allerdings zum Teil erheblich, so daß eine einfache und einheitliche Umsetzung in Programme nicht direkt möglich ist. Aus diesem Grunde wurden alle Reihen zunächst mit Hilfe eines Formelmanipulationsprogrammes bearbeitet. Dabei wurden die Keplerbewegung und die Störsterme unabhängig von der ursprünglichen Darstellung als periodische Bewegung in den Koordinaten Länge, Breite und Radius formuliert. Zusätzlich wurde eine Vielzahl von Termen ausgeschieden, die aufgrund ihrer Größe (meist $\ll 1''$) keinen Einfluß auf die beobachtete Genauigkeit der Reihen haben. Diese Umarbeitung erlaubt eine vergleichsweise kurze und kompakte Programmierung, ohne die Genauigkeit der Entwicklungen nennenswert zu verschlechtern.

6.1 Reihenentwicklung des Keplerproblems

Die Bahn eines Planeten in einer ungestörten Keplerellipse ist eine streng periodische Funktion der mittleren Anomalie M . Wenn sich die mittlere Anomalie um 360° ändert, dann befindet sich der Planet nach einem Umlauf wieder am selben Ort. Jede Funktion des Bahnortes läßt sich deshalb als Fourierreihe, also als Summe der Winkelfunktionen Sinus und Cosinus von Vielfachen des Argumentes M darstellen. Beispiele hierfür sind die Reihenentwicklungen der Differenz zwischen wahrer und mittlerer Anomalie

$$\nu - M = a_1 \sin(M) + a_2 \sin(2M) + a_3 \sin(3M) + \dots$$

und der Entfernung r in Einheiten der großen Halbachse a :

$$r/a = b_0 + b_1 \cos(M) + b_2 \cos(2M) + \dots .$$

Die Koeffizienten a_i und b_i dieser Reihen sind Funktionen der Bahnexzentrizität e . Die wichtigsten Terme für $\nu - M$ (die sogenannte *Mittelpunktsgleichung*) und r/a lauten bis zur Ordnung e^2 :

$$\nu - M = 2e \sin(M) + \frac{5}{4}e^2 \sin(2M) + \dots \quad (\text{Bogenmaß}) \quad (6.1)$$

und

$$r/a = \left(1 + \frac{1}{2}e^2\right) - e \cos(M) - \frac{1}{2}e^2 \cos(2M) + \dots . \quad (6.2)$$

Beide Reihen konvergieren für kleine Exzentrizitäten sehr rasch, weil die einzelnen Glieder von Term zu Term größtenteils um den Faktor e kleiner werden. Für die Erdbahn mit $e = 0.0167$ ist etwa

$$\begin{aligned} \nu &= M + 0.0334 \sin(M) + 0.00035 \sin(2M) + \dots \quad (\text{Bogenmaß}) \\ &= M + 1.916 \sin(M) + 0.020 \sin(2M) + \dots \\ r &= a \cdot (1.00014 - 0.0167 \cos(M) - 0.00014 \cos(2M) + \dots) . \end{aligned}$$

Der Fehler dieser kurzen Formel beträgt nur rund $1''$ bzw. 10^{-5} AE und lässt sich auch noch weiter verringern, wenn man zusätzliche Terme hinzunimmt.

Da die vollständige Herleitung der obigen Formeln etwas tiefergehende Kenntnisse verlangt, soll hier nur das erste Glied der Mittelpunktsgleichung hergeleitet werden. Hierzu sei noch einmal an die Keplergleichung (4.7)

$$E - e \sin E = M$$

erinnert. Da E und M sich nur um einen Term der Ordnung e unterscheiden, kann man in guter Näherung $e \sin E$ durch $e \sin M$ ersetzen und erhält zunächst

$$E - M \approx e \sin M$$

für die Differenz zwischen der exzentrischen und der wahren Anomalie. Weiter folgt aus der Kegelschnittsgleichung $r = a(1 - e^2)/(1 + e \cos \nu)$ mit $r \cos \nu = a(\cos E - e)$ die Beziehung

$$\cos \nu \approx \cos E - e(1 - \cos \nu \cos E) .$$

Im zweiten Term der rechten Seite darf man nun wieder $\cos \nu$ durch $\cos E$ ersetzen und gelangt so zu

$$\cos \nu \approx \cos E - \{e \sin E\} \sin E .$$

Vergleicht man dies mit der Taylorentwicklung der Cosinusfunktion,

$$\cos(x + \Delta x) \approx \cos x - \Delta x \sin x ,$$

dann erkennt man, daß für kleine Exzentrizitäten $\nu - E$ ungefähr $e \sin E$ ist. Damit gilt also für das erste Glied der Mittelpunktsgleichung wie behauptet

$$\nu - M = (\nu - E) + (E - M) \approx e \sin E + e \sin M \approx 2e \sin M .$$

Für kleine Bahnneigungen i verläuft die Bahn immer in der Nähe der Ekliptik. Daher unterscheidet sich die ekliptikale Länge l eines Planeten nur um eine kleine Korrektur R (*Reduktion auf die Ekliptik*) von der Summe $\varpi + \nu$ aus Perihellänge und wahrer Anomalie. R verschwindet für $i = 0$ und hat für kleine Bahnneigungen in erster Näherung den Wert

$$R = l - (\nu + \varpi) \approx -\tan^2\left(\frac{i}{2}\right) \cdot \sin(2u) \quad (\text{Bogenmaß}) \quad (6.3)$$

mit $u = \nu + \omega = \nu + \varpi - \Omega$. Wegen $\nu \approx M$ ($u \approx M + \omega$) lässt sich R auch in der Form

$$R \approx -\tan^2\left(\frac{i}{2}\right) \cdot \{\sin(2\omega)\cos(2M) + \cos(2\omega)\sin(2M)\}$$

schreiben. Den Wert von R im Gradmaß erhält man durch Multiplikation mit $180^\circ/\pi$.

Schließlich kann man noch die ekliptikale Breite b des Planeten in ähnlicher Weise behandeln. Die Gleichung (4.20)

$$z = r \cdot (\sin(\nu + \omega) \sin(i)) \quad (6.4)$$

für die z -Koordinate als Funktion der Bahnelemente führt mit

$$\sin(b) = z/r \quad (6.5)$$

für kleine Bahnneigungen i und ekliptikale Breiten b zu der Näherung

$$b \approx i \sin(\nu + \omega) = \{i \sin \omega\} \cos(\nu) + \{i \cos \omega\} \sin(\nu)$$

Ersetzt man ν durch $\nu \approx M + 2e \sin M$, dann folgt durch Taylorentwicklung

$$\cos \nu \approx \cos M - (2e \sin M) \sin M = \cos(M) - e(1 - \cos(2M))$$

$$\sin \nu \approx \sin M + (2e \sin M) \cos M = \sin(M) + e \sin(2M)$$

und somit

$$b \approx \{i \sin \omega\} \cdot \{\cos(M) - e(1 - \cos(2M))\} + \\ \{i \cos \omega\} \cdot \{\sin(M) + e \sin(2M)\} .$$

Die wichtigsten Glieder einer Darstellung ungestörter Planetenbahnen durch trigonometrische Reihen lauten damit:

$$\begin{aligned} l &= \varpi + M \\ &+ \frac{180^\circ}{\pi} \cdot \{2e\} \cdot \sin(M) \\ &+ \frac{180^\circ}{\pi} \cdot \left\{ \frac{5}{4}e^2 - \tan^2\left(\frac{i}{2}\right) \cos(2\omega) \right\} \cdot \sin(2M) \\ &+ \frac{180^\circ}{\pi} \cdot \left\{ -\tan^2\left(\frac{i}{2}\right) \sin(2\omega) \right\} \cdot \cos(2M) \end{aligned} \quad (6.6)$$

$$\begin{aligned} b &= -\{ie \sin \omega\} \\ &+ \{i \sin \omega\} \cos(M) + \{i \cos \omega\} \sin(M) \\ &+ \{ie \sin \omega\} \cos(2M) + \{ie \cos \omega\} \sin(2M) \end{aligned} \quad (6.7)$$

$$r = \{a(1 + e^2/2)\} - \{ae\} \cdot \cos(M) - \{ae^2/2\} \cdot \cos(2M) . \quad (6.8)$$

Setzt man in diese Gleichungen konkrete Zahlenwerte für die Bahnelemente ein, dann erhält man sehr kompakte Formeln für die ekliptikalnen Koordinaten (l, b, r)

als Funktion der mittleren Anomalie M . Für Jupiter ergeben sich zum Beispiel mit

$$\begin{aligned} a &= 5.2 \text{AE} & \Omega &= 100^\circ 0' & \omega &= 274^\circ 0' \\ e &= 0.048 & i &= 1^\circ 31' & \varpi &= 14^\circ 0' \end{aligned}$$

die folgenden Reihen:

$$\begin{aligned} l &= M + 14^\circ 0' + 19800'' \sin(M) + 620'' \sin(2M) + 4'' \cos(2M) \\ b &= +226'' - 4700'' \cos(M) + 329'' \sin(M) - 226'' \cos(2M) + 16'' \sin(2M) \\ r &= (5.206 - 0.250 \cos(M) - 0.006 \cos(2M)) \text{ AE} \quad . \end{aligned}$$

6.2 Störungsterme

Die gegenseitigen Störungen der Planeten, die für die Abweichungen von einer Keplerbahn verantwortlich sind, schlagen sich in zusätzlichen Termen der Reihenentwicklungen nieder, die neben der Anomalie des gestörten Planeten auch die des störenden Planeten enthalten. Im folgenden bezeichnet M_n die mittlere Anomalie des n -ten Planeten, also etwa M_5 die des Jupiter und M_6 die des Saturn:

$$\begin{aligned} M_1 &= 0^\circ 4855407 + 415^\circ 2014314 \cdot T && (\text{Merkur}) \\ M_2 &= 0^\circ 1400197 + 162^\circ 5494552 \cdot T && (\text{Venus}) \\ M_3 &= 0^\circ 9931266 + 99^\circ 9973604 \cdot T && (\text{Sonne, Erde}) \\ M_4 &= 0^\circ 0538553 + 53^\circ 1662736 \cdot T && (\text{Mars}) \\ M_5 &= 0^\circ 0565314 + 8^\circ 4302963 \cdot T && (\text{Jupiter}) \\ M_6 &= 0^\circ 8829867 + 3^\circ 3947688 \cdot T && (\text{Saturn}) \\ M_7 &= 0^\circ 3967117 + 1^\circ 1902849 \cdot T && (\text{Uranus}) \\ M_8 &= 0^\circ 7214906 + 0^\circ 6068526 \cdot T && (\text{Neptun}) \\ M_9 &= 0^\circ 0385795 + 0^\circ 4026667 \cdot T && (\text{Pluto}) \quad . \end{aligned} \tag{6.9}$$

Die Werte sind hier in Vielfachen eines Umlaufs (1°) angegeben¹. Multipliziert man sie mit 360° oder 2π , dann erhält man die mittleren Anomalien im gewohnten Grad- oder Bogenmaß. Die Zeit

$$T = (\text{JD} - 2451545)/36525$$

wird wie gewohnt in julianischen Jahrhunderten seit der Epoche J2000 gezählt.

Die verschiedenen Störungsterme lassen sich am übersichtlichsten in Form einer Tabelle darstellen, wie sie in Tabelle 6.1 für den Planeten Jupiter angegeben ist. Darin sind sämtliche periodischen Variationen der ekliptikalnen Länge (dl) und Breite (db) sowie der Entfernung von der Sonne (dr) aufgeführt. Terme mit gleichen Argumenten der trigonometrischen Funktionen sind jeweils in einer Zeile zusammengefaßt, was für die spätere Auswertung von Nutzen ist. Ausgeschrieben

¹Die in den Programmen verwendeten Zahlen weichen in einzelnen Fällen um kleine Beträge von den genannten Werten ab. Dies ist kein Fehler, sondern eine Folge der unterschiedlichen Theorien, die den Reihenentwicklungen zugrunde liegen.

Tabelle 6.1. Periodische Terme der Jupiterbahn

i_5	i_T	$dl["]$		$dr[10^{-5} \text{ AE}]$		$db["]$		Keplerterme	
		cos	sin	cos	sin	cos	sin		
1	0	-113.1	19998.6	-25208.2	-142.2	-4670.7	288.9		
1	1	-76.1	66.9	-84.2	-95.8	21.6	29.4		
1	2	-0.5	-0.3	0.4	-0.7	0.1	-0.1		
2	0	-3.4	632.0	-610.6	-6.5	-226.8	12.7		
2	1	-4.2	3.8	-4.1	-4.5	0.2	0.6		
3	0	-0.1	28.0	-22.1	-0.2	-12.5	0.7		
4	0	0.0	1.4	-1.0	0.0	-0.6	0.0		
i_5	i_6	i_T							
-1	-1	0	-0.2	1.4	2.0	0.6	0.1	-0.2	Saturn
0	-1	0	9.4	8.9	3.9	-8.3	-0.4	-1.4	
0	-2	0	5.6	-3.0	-5.4	-5.7	-2.0	0.0	
0	-3	0	-4.0	-0.1	0.0	5.5	0.0	0.0	
0	-5	0	3.3	-1.6	-1.6	-3.1	-0.5	-1.2	
1	-1	0	78.8	-14.5	11.5	64.4	-0.2	0.2	
1	-2	0	-2.0	-132.4	28.8	4.3	-1.7	0.4	
1	-2	1	-1.1	-0.7	0.2	-0.3	0.0	0.0	
1	-3	0	-7.5	-6.8	-0.4	-1.1	0.6	-0.9	
1	-4	0	0.7	0.7	0.6	-1.1	0.0	-0.2	
1	-5	0	51.5	-26.0	-32.5	-64.4	-4.9	-12.4	
1	-5	1	-1.2	-2.2	-2.7	1.5	-0.4	0.3	
2	-1	0	5.3	-0.7	0.7	6.1	0.2	1.1	
2	-2	0	-76.4	-185.1	260.2	-108.0	1.6	0.0	
2	-3	0	66.7	47.8	-51.4	69.8	0.9	0.3	
2	-3	1	0.6	-1.0	1.0	0.6	0.0	0.0	
2	-4	0	17.0	1.4	-1.8	9.6	0.0	-0.1	
2	-5	0	1066.2	-518.3	-1.3	-23.9	1.8	-0.3	
2	-5	1	-25.4	-40.3	-0.9	0.3	0.0	0.0	
2	-5	2	-0.7	0.5	0.0	0.0	0.0	0.0	
3	-2	0	-5.0	-11.5	11.7	-5.4	2.1	-1.0	
3	-3	0	16.9	-6.4	13.4	26.9	-0.5	0.8	
3	-4	0	7.2	-13.3	20.9	10.5	0.1	-0.1	
3	-5	0	68.5	134.3	-166.9	86.5	7.1	15.2	*
3	-5	1	3.5	-2.7	3.4	4.3	0.5	-0.4	
3	-6	0	0.6	1.0	-0.9	0.5	0.0	0.0	
3	-7	0	-1.1	1.7	-0.4	-0.2	0.0	0.0	
4	-2	0	-0.3	-0.7	0.4	-0.2	0.2	-0.1	
4	-3	0	1.1	-0.6	0.9	1.2	0.1	0.2	
4	-4	0	3.2	1.7	-4.1	5.8	0.2	0.1	
4	-5	0	6.7	8.7	-9.3	8.7	-1.1	1.6	
4	-6	0	1.5	-0.3	0.6	2.4	0.0	0.0	
4	-7	0	-1.9	2.3	-3.2	-2.7	0.0	-0.1	
4	-8	0	0.4	-1.8	1.9	0.5	0.0	0.0	
4	-9	0	-0.2	-0.5	0.3	-0.1	0.0	0.0	
4	-10	0	-8.6	-6.8	-0.4	0.1	0.0	0.0	
4	-10	1	-0.5	0.6	0.0	0.0	0.0	0.0	
5	-5	0	-0.1	1.5	-2.5	-0.8	-0.1	0.1	
5	-6	0	0.1	0.8	-1.6	0.1	0.0	0.0	
5	-9	0	-0.5	-0.1	0.1	-0.8	0.0	0.0	
5	-10	0	2.5	-2.2	2.8	3.1	0.1	-0.2	
i_5	i_6	i_T							
1	-1	0	0.4	0.9	0.0	0.0	0.0	0.0	Uranus
1	-2	0	0.4	0.4	-0.4	0.3	0.0	0.0	
i_5	i_6	i_7							
2	-6	3	-0.8	8.5	-0.1	0.0	0.0	0.0	Saturn und Uranus
3	-6	3	0.4	0.5	-0.7	0.5	-0.1	0.0	

bedeutet die mit einem Stern (*) markierte Zeile zum Beispiel, daß zu Länge, Breite und Radius die Werte

$$\begin{aligned} dl &= 68.^{\circ}5 \cdot \cos(3M_5 - 5M_6) + 134.^{\prime\prime}3 \cdot \sin(3M_5 - 5M_6) \\ dr &= (-166.9 \cdot \cos(3M_5 - 5M_6) + 86.5 \cdot \sin(3M_5 - 5M_6)) \cdot 10^{-5} \text{ AE} \\ db &= 7.^{\prime\prime}1 \cdot \cos(3M_5 - 5M_6) + 15.^{\prime\prime}2 \cdot \sin(3M_5 - 5M_6) \end{aligned}$$

hinzu zu addieren sind. Terme, die zusätzlich mit der Zeit T oder mit T^2 zu multiplizieren sind, sind durch eine 1 oder 2 in der mit i_T überschriebenen Spalte gekennzeichnet.

Der Einfluß der anderen Planeten macht sich aber nicht nur in periodischen Störungen sondern auch in einer langfristigen (*säkularen*) Veränderung der mittleren Bahnelemente bemerkbar. Dies hat zur Folge, daß nun auch in den Termen, die eigentlich die reine Keplerbewegung beschreiben, Glieder auftauchen, die von der Zeit T abhängen. Dies sei noch einmal am Beispiel der Jupiterbahn illustriert. In der Funktion `JupiterPos` werden die folgenden Gleichungen zur Berechnung der Jupiterkoordinaten verwendet:

$$\begin{aligned} l &= M_5 + 14^\circ 000076 + (5025.^{\prime\prime}2 + 0.^{\prime\prime}8T)T + dl \\ b &= 227.^{\circ}3 - 0.^{\prime\prime}3T + db \\ r &= (5.208873 + 0.000041T)\text{AE} + dr \end{aligned} .$$

dl , db und dr beinhalten darin sämtliche periodischen Anteile der Bahn, also die Keplerterme und die Störungen durch Saturn und Uranus. Die so bestimmten Werte der ekliptikalnen Länge und Breite beziehen sich auf das *mittlere Äquinoktium des Datums*. Die Formel für l enthält deshalb den auffällig großen säkularen Term $+5025.^{\prime\prime}2T$, der die Wanderung des Frühlingspunktes wiedergibt.

Aus Platzgründen können hier nicht alle Störungsterme für sämtliche Planeten wiedergegeben werden. Wegen der einheitlichen Struktur der einzelnen Reihenentwicklungen sollten die Ausführungen über die Jupiterbahn aber genügen, um anhand der Programme die jeweils verwendeten Formeln und Terme erkennen zu können.

Unterschiede zu den übrigen Planeten ergeben sich allerdings bei der Behandlung der Erdbahn. Die Erde umkreist zusammen mit dem Mond die Sonne, wobei man sich beide Massen im gemeinsamen Schwerpunkt vereinigt denken kann. Die Bewegung dieses Schwerpunkts, die im übrigen die mittlere Ekliptik definiert, läßt sich wie die der anderen Planeten als gestörte Keplerbahn auffassen und durch die besprochenen Reihenentwicklungen darstellen. Entsprechend dem Verhältnis von Mond- und Erdmasse beträgt die Entfernung des Erdmittelpunktes vom Schwerpunkt 1/81 der Entfernung des Mondes von der Erde und damit rund 3/4 Erdradien (vgl. Abb. 6.1). Von der Sonne aus gesehen schwankt die ekliptikale Länge der Erde während eines Monats deshalb mit einer Auslenkung von $7.^{\prime\prime}$ um ihre mittlere Lage. Gleichzeitig variiert die ekliptikale Breite der Erde um $0.^{\prime\prime}5$.

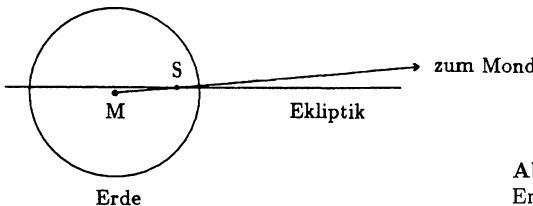


Abb. 6.1. Zur Lage des Schwerpunkts von Erde und Mond

6.3 Numerische Behandlung der Reihenentwicklungen

Um Genauigkeiten im Bogensekundenbereich zu ermöglichen, benötigt man für jeden Planeten mehrere hundert Störungsterme. Würde man diese explizit programmieren, dann erhielte man nicht nur einen sehr langen und unübersichtlichen Programmtext, sondern müßte auch unnötig lange Rechenzeiten in Kauf nehmen. Dies hängt damit zusammen, daß die Auswertung trigonometrischer Funktionen im Vergleich zu elementaren Rechenoperationen wie Addition und Multiplikation relativ aufwendig und langsam ist. Angesichts der großen Zahl von Winkelfunktionen in den Reihenentwicklungen macht sich dies auch bei schnellen Rechnern durchaus bemerkbar.

Durch Anwendung der Additionstheoreme für Sinus und Cosinus kann man sich jedoch einen Großteil der Arbeit ersparen. Sie dienen dazu, die Winkelfunktionen einer Summe oder Differenz von Winkeln zu berechnen, wenn die Funktionen der Einzelwinkel bekannt sind:

$$\begin{aligned}\cos(\alpha_1 + \alpha_2) &= \cos \alpha_1 \cos \alpha_2 - \sin \alpha_1 \sin \alpha_2 \\ \sin(\alpha_1 + \alpha_2) &= \sin \alpha_1 \cos \alpha_2 + \cos \alpha_1 \sin \alpha_2\end{aligned}. \quad (6.10)$$

Man kann damit sehr bequem $\cos(iM)$ und $\sin(iM)$ für Vielfache $i = 2, 3, \dots$ der mittleren Anomalie M eines Planeten aus $\cos M$ und $\sin M$ berechnen und zwischenspeichern. Hierzu benötigt man zunächst die Funktion AddThe:

```
//-----
// AddThe: Wertet cos(alpha+beta) und sin(alpha+beta) mit Hilfe der Additions-
//          theoreme aus
//      c1,s1    cos(alpha), sin(alpha)
//      c2,s2    cos(beta), sin(beta)
//      c,s      cos(alpha+beta), sin(alpha+beta)
//-----
void AddThe ( double c1, double s1, double c2, double s2,
              double& c, double& s )
{
    c = c1 * c2 - s1 * s2;
    s = s1 * c2 + c1 * s2;
}
```

Alle weiteren Operationen sowie die notwendigen Hilfsfelder zur Speicherung der vorberechneten Größen sind in einer eigenen Klasse Pert zusammengefaßt. Sie stellt im wesentlichen die beiden Elementfunktionen Init und Term bereit. Bezeichnet man mit M die mittlere Anomalie des gestörten Planeten und mit m die mittlere Anomalie des störenden Planeten, dann berechnet Init die Größen

$\sin(IM + \phi)$ und $\cos(IM + \phi)$ für Vielfache $I_{\min} \leq I \leq I_{\max}$ sowie die Größen $\sin(im)$ und $\cos(im)$ für Vielfache $i_{\min} \leq i \leq i_{\max}$. Der Winkel ϕ ist dabei eine Hilfsgröße, die es ermöglicht auch Störungsterme, die von den mittleren Anomalien dreier Planeten abhängen, in einheitlicher Weise zu behandeln. Bei der Auswertung eines einzelnen Störungsterms greift die Methode `Term` auf die vorberechneten Winkelfunktionen zu und summiert die Beiträge zu Länge, Radius und Breite in entsprechenden Elementen auf. Die akkumulierten Störungsterme können dann über die Methoden `dl`, `dr` und `db` abgefragt werden.

```

// Konstanten
const int o    = 16;           // Index-Versatz
const int dim = 2*o+1;         // Dimension eines Arbeits-Feldes

// Definition der Klasse Pert zum Aufsummieren von trigonometrischen
// Stoerungsreihen
class Pert
{
public:
    // Setze Zeit, mittlere Anomalien und Indexbereich
    void Init ( double T,
                double M, int I_min, int I_max,
                double m, int i_min, int i_max,
                double phi=0.0 );
    // Summation der Stoerungen in Laenge, Entfernung und Breite
    void Term ( int I, int i, int iT,
                 double dlc, double dls,
                 double drc, double drs,
                 double dbc, double dbs );
    // Ermittle die Stoerungen in Laenge, Entfernung und Breite
    double dl();
    double dr();
    double db();
private:
    double m_T, m_cosM, m_sinM, m_C[dim], m_S[dim], m_c[dim], m_s[dim];
    double m_dl, m_db, m_dr, m_u, m_v;
};

// Setze Zeit, mittlere Anomalien und Indexbereich
void Pert::Init ( double T,
                  double M, int I_min, int I_max,
                  double m, int i_min, int i_max,
                  double phi )
{
    int i;
    m_dl=0.0; m_dr=0.0; m_db=0.0; // Stoerungen auf Null setzen
    m_T=T;                                // Setze Zeit
    // Kosinus und Sinus von Vielfachen von M
    m_cosM=cos(M); m_sinM=sin(M);
    m_C[o]=cos(phi);
    m_S[o]=sin(phi);
    for (i=0; i<I_max; i++)
        AddThe ( m_C[o+i],m_S[o+i], +m_cosM,+m_sinM, m_C[o+i+1],m_S[o+i+1] );
    for (i=0; i>I_min; i--)
        AddThe ( m_C[o+i],m_S[o+i], +m_cosM,-m_sinM, m_C[o+i-1],m_S[o+i-1] );
}

```

```

// Kosinus und Sinus von Vielfachen von m
m_c[o]=1.0; m_c[o+1]=cos(m); m_c[o-1]=-m_c[o+1];
m_s[o]=0.0; m_s[o+1]=sin(m); m_s[o-1]=-m_s[o+1];
for (i=1; i<i_max; i++)
    AddThe ( m_c[o+i],m_s[o+i], m_c[o+1],m_s[o+1], m_c[o+i+1],m_s[o+i+1] );
for (i=-1; i>i_min; i--)
    AddThe ( m_c[o+i],m_s[o+i], m_c[o-1],m_s[o-1], m_c[o+i-1],m_s[o+i-1] );
}

// Summation der Stoerungen in Laenge, Entfernung und Breite
void Pert::Term ( int I, int i, int iT,
                  double dlc, double dls,
                  double drc, double drs,
                  double dbc, double dbs )
{
    if (iT == 0)
        AddThe ( m_C[o+I],m_S[o+I], m_c[o+i],m_s[o+i], m_u,m_v );
    else
        { m_u *= m_T; m_v *= m_T; };
    m_dl += ( dlc*m_u + dls*m_v );
    m_dr += ( drc*m_u + drs*m_v );
    m_db += ( dbc*m_u + dbs*m_v );
}

// Ermittle die Stoerungen in Laenge, Entfernung und Breite
double Pert::dl() { return m_dl; }
double Pert::dr() { return m_dr; }
double Pert::db() { return m_db; }

```

Das nachfolgende Beispiel der Funktion `JupiterPos` zeigt, wie sich nach diesem Schema sämtliche Störungsterme auswerten und aufsummieren lassen. Die Methode `Term` berechnet zu den Indizes i_5 und i_j die Werte $u = \cos(i_5 M_5 + i_j M_j)$ und $v = \sin(i_5 M_5 + i_j M_j)$, multipliziert diese mit den entsprechenden Koeffizienten und addiert sie schließlich zu dl , dr und db . Unter der Voraussetzung, daß alle Terme mit gleichem i_5 und i_j , aber verschiedenem $i_T = 0, 1, \dots$ hintereinander ausgewertet werden, genügt es, wie in `Term` verwirklicht, u und v nur bei $i_T = 0$ zu berechnen und für $i_T = 1$ oder $i_T = 2$ jeweils einmal mit T zu multiplizieren. Der auf den ersten Blick etwas hohe Aufwand macht sich sehr schnell bezahlt, wenn man – wie im vorliegenden Fall – mit sehr vielen Termen arbeitet.

```

//-----
// JupiterPos: Position von Jupiter aus analytischer Reihenentwicklung
//   T          Zeit in Julianischen Jahrhunderten seit J2000
//   <return>   Heliozentrischer Ort [AE], Ekliptik und Aequinoktium des Datums
//-----
Vec3D JupiterPos (double T)
{
    double M5,M6,M7;           // Mittlere Anomalien
    Pert Sat,Ura;             // Stoerungen
    double phi,c,s;
    double dl, dr, db;         // Korrekturen in Laenge ["],
                                // Entfernung [AE] und Breite ["]
    double l,b,r;              // Ekliptikale Koordinaten

```

```

// Mittlere Anomalien der Planeten in [rad]
M5 = pi2 * Frac ( 0.0565314 + 8.4302963*T );
M6 = pi2 * Frac ( 0.8829867 + 3.3947688*T );
M7 = pi2 * Frac ( 0.3969537 + 1.1902586*T );

// Keplerbewegung und Stoerungen durch Saturn
Sat.Init ( T, M5,-1,5, M6,-10,0 );
Sat.Term (-1, -1,0, -0.2, 1.4, 2.0, 0.6, 0.1, -0.2);
Sat.Term ( 0, -1,0, 9.4, 8.9, 3.9, -8.3, -0.4, -1.4);
Sat.Term ( 0, -2,0, 5.6, -3.0, -5.4, -5.7, -2.0, 0.0);
Sat.Term ( 0, -3,0, -4.0, -0.1, 0.0, 5.5, 0.0, 0.0);
Sat.Term ( 0, -5,0, 3.3, -1.6, -1.6, -3.1, -0.5, -1.2);
Sat.Term ( 1, 0,0,-113.1,19998.6,-25208.2,-142.2,-4670.7,288.9);
Sat.Term ( 1, 0,1, -76.1, 66.9, -84.2, -95.8, 21.6, 29.4);
Sat.Term ( 1, 0,2, -0.5, -0.3, 0.4, -0.7, 0.1, -0.1);
Sat.Term ( 1, -1,0, 78.8, -14.5, 11.5, 64.4, -0.2, 0.2);
Sat.Term ( 1, -2,0, -2.0, -132.4, 28.8, 4.3, -1.7, 0.4);
Sat.Term ( 1, -2,1, -1.1, -0.7, 0.2, -0.3, 0.0, 0.0);
Sat.Term ( 1, -3,0, -7.5, -6.8, -0.4, -1.1, 0.6, -0.9);
Sat.Term ( 1, -4,0, 0.7, 0.7, 0.6, -1.1, 0.0, -0.2);
Sat.Term ( 1, -5,0, 51.5, -26.0, -32.5, -64.4, -4.9,-12.4);
Sat.Term ( 1, -5,1, -1.2, -2.2, -2.7, 1.5, -0.4, 0.3);
Sat.Term ( 2, 0,0, -3.4, 632.0, -610.6, -6.5, -226.8, 12.7);
Sat.Term ( 2, 0,1, -4.2, 3.8, -4.1, -4.5, 0.2, 0.6);
Sat.Term ( 2, -1,0, 5.3, -0.7, 0.7, 6.1, 0.2, 1.1);
Sat.Term ( 2, -2,0, -76.4, -185.1, 260.2,-108.0, 1.6, 0.0);
Sat.Term ( 2, -3,0, 66.7, 47.8, -51.4, 69.8, 0.9, 0.3);
Sat.Term ( 2, -3,1, 0.6, -1.0, 1.0, 0.6, 0.0, 0.0);
Sat.Term ( 2, -4,0, 17.0, 1.4, -1.8, 9.6, 0.0, -0.1);
Sat.Term ( 2, -5,0,1066.2, -518.3, -1.3, -23.9, 1.8, -0.3);
Sat.Term ( 2, -5,1, -25.4, -40.3, -0.9, 0.3, 0.0, 0.0);
Sat.Term ( 2, -5,2, -0.7, 0.5, 0.0, 0.0, 0.0, 0.0);
Sat.Term ( 3, 0,0, -0.1, 28.0, -22.1, -0.2, -12.5, 0.7);
Sat.Term ( 3, -2,0, -5.0, -11.5, 11.7, -5.4, 2.1, -1.0);
Sat.Term ( 3, -3,0, 16.9, -6.4, 13.4, 26.9, -0.5, 0.8);
Sat.Term ( 3, -4,0, 7.2, -13.3, 20.9, 10.5, 0.1, -0.1);
Sat.Term ( 3, -5,0, 68.5, 134.3, -166.9, 86.5, 7.1, 15.2);
Sat.Term ( 3, -5,1, 3.5, -2.7, 3.4, 4.3, 0.5, -0.4);
Sat.Term ( 3, -6,0, 0.6, 1.0, -0.9, 0.5, 0.0, 0.0);
Sat.Term ( 3, -7,0, -1.1, 1.7, -0.4, -0.2, 0.0, 0.0);
Sat.Term ( 4, 0,0, 0.0, 1.4, -1.0, 0.0, -0.6, 0.0);
Sat.Term ( 4, -2,0, -0.3, -0.7, 0.4, -0.2, 0.2, -0.1);
Sat.Term ( 4, -3,0, 1.1, -0.6, 0.9, 1.2, 0.1, 0.2);
Sat.Term ( 4, -4,0, 3.2, 1.7, -4.1, 5.8, 0.2, 0.1);
Sat.Term ( 4, -5,0, 6.7, 8.7, -9.3, 8.7, -1.1, 1.6);
Sat.Term ( 4, -6,0, 1.5, -0.3, 0.6, 2.4, 0.0, 0.0);
Sat.Term ( 4, -7,0, -1.9, 2.3, -3.2, -2.7, 0.0, -0.1);
Sat.Term ( 4, -8,0, 0.4, -1.8, 1.9, 0.5, 0.0, 0.0);
Sat.Term ( 4, -9,0, -0.2, -0.5, 0.3, -0.1, 0.0, 0.0);
Sat.Term ( 4,-10,0, -8.6, -6.8, -0.4, 0.1, 0.0, 0.0);
Sat.Term ( 4,-10,1, -0.5, 0.6, 0.0, 0.0, 0.0, 0.0);
Sat.Term ( 5, -5,0, -0.1, 1.5, -2.5, -0.8, -0.1, 0.1);
Sat.Term ( 5, -6,0, 0.1, 0.8, -1.6, 0.1, 0.0, 0.0);
Sat.Term ( 5, -9,0, -0.5, -0.1, 0.1, -0.8, 0.0, 0.0);

```

```

Sat.Term ( 5,-10,0,    2.5,   -2.2,     2.8,    3.1,     0.1, -0.2);
dl = Sat.dl(); dr = Sat.dr(); db = Sat.db();

// Stoerungen durch Uranus
Ura.Init ( T, M5,1,1, M7,-2,-1 );
Ura.Term ( 1, -1,0,    0.4,     0.9,     0.0,    0.0,     0.0,  0.0 );
Ura.Term ( 1, -2,0,    0.4,     0.4,    -0.4,    0.3,     0.0,  0.0 );
dl += Ura.dl(); dr += Ura.dr(); db += Ura.db();

// Stoerungen durch Saturn und Uranus
phi = (2*M5-6*M6+3*M7); c=cos(phi); s=sin(phi);
dl += -0.8*c+8.5*s; dr += -0.1*c;
phi = (3*M5-6*M6+3*M7); c=cos(phi); s=sin(phi);
dl += +0.4*c+0.5*s; dr += -0.7*c+0.5*s; db += -0.1*c;

// Ekliptikale Koordinaten ([rad],[AE])
l = pi2 * Frac (0.0388910 + M5/pi2 + ((5025.2+0.8*T)*T + dl) / 1296.0e3);
r = 5.208873 + 0.000041*T + dr * 1.0E-5;
b = (227.3 - 0.3*T + db) / Arcs;
return Vec3D ( Polar(l,b,r) ); // Ort
}

```

JupiterPos und die analog aufgebauten Funktionen MercuryPos, VenusPos, ..., PlutoPos liefern zu gegebener Zeit T die heliozentrischen Koordinaten Länge l , Breite b und Radius r des jeweiligen Planeten in Bezug auf Ekliptik und Frühlingspunkt des Datums. Die Zeit wird in julianischen Jahrhunderten ab der Epoche J2000 gemessen:

$$T = (\text{JD} - 2451545) / 36525 .$$

Bei der Berechnung des julianischen Datums JD muß darauf geachtet werden, daß nicht die Weltzeit UT, sondern die Ephemeridenzeit ET (beziehungsweise die dynamische Zeit TDT/TDB) als Uhrzeit eingesetzt wird (vergleiche hierzu Abschn. 3.4). Eine entsprechende Funktion zur Berechnung geozentrischer Sonnenkoordinaten wurde bereits in Kap. 2 vorgestellt.

Die Routine PlutoPos unterscheidet sich im Aufbau von den übrigen Funktionen. Die Koordinaten werden zunächst relativ zur festen Ekliptik von 1950 berechnet und anschließend in das Äquinoktium des Datums transformiert. Diese Vorgehensweise ist aufgrund der hohen Bahnneigung des Planeten Pluto nötig. Eine vollständige Entwicklung der Koordinaten würde zu einer Vielzahl säkularer Glieder führen. Weiterhin ist zu beachten, daß PlutoPos nur für Berechnungen zwischen den Jahren 1890 und 2100 verwendet werden kann. Die Ursache hierfür liegt darin, daß die verwendete Reihenentwicklung nicht aus einer Störungstheorie, sondern durch Fourieranalyse einer numerisch integrierten Ephemeride über den genannten Zeitraum gewonnen wurde. Der Fehler der berechneten Koordinaten wächst bereits wenige Jahre vor 1890 oder nach 2100 stark an und erreicht dabei Werte von über $0^\circ 5$.

Zum einfacheren Gebrauch sind die einzelnen Routinen noch einmal in einer übergeordneten Funktion PertPosition zusammengefaßt. Analog zu der bereits vorgestellten Funktion KepPosition können damit heliozentrische Koordinaten für alle Planeten in einer einzigen Funktion berechnet werden.

```

//-----
// PertPosition: Berechnung des Planetenortes aus analytischen Reihenent-
//                  wicklungen
// Planet      Identifiziert den Planeten
// T           Zeit in Julianischen Jahrhunderten seit J2000
// <return>    Heliozentrischer Ort [AE], Ekliptik und Aequinoktium des Datums
//-----
Vec3D PertPosition (PlanetType Planet, double T)
{
    Vec3D r; // Nullvektor;
    switch ( Planet )
    {
        case Sun:      break;
        case Mercury: r = MercuryPos(T);   break;
        case Venus:    r = VenusPos(T);    break;
        case Earth:    r = (-SunPos(T));   break;
        case Mars:     r = MarsPos(T);    break;
        case Jupiter:  r = JupiterPos(T); break;
        case Saturn:   r = SaturnPos(T);  break;
        case Uranus:   r = UranusPos(T);  break;
        case Neptune:  r = NeptunePos(T); break;
        case Pluto:    r = PlutoPos(T);
    }
    return r;
}

```

Die Auswahl des Planeten erfolgt dabei über einen Parameter `Planet` vom Typ

```
enum PlanetType { Sun, Mercury, Venus, Earth, Mars,
                 Jupiter, Saturn, Uranus, Neptune, Pluto };
```

der auch direkt auf einen Schleifenzähler mit den Werten 0 bis 9 abgebildet werden kann.

Tabelle 6.2. Mittlere Fehler der Programme `MercuryPos...``PlutoPos` für den Zeitraum der Jahre 1750–2250 bzw. 1890–2100 (Pluto)

Planet	$\Delta l ["]$	$\Delta b ["]$	$\Delta r [AE]$
Merkur	1.0-1.5	0.7-1.0	$1.0 \cdot 1.5 \cdot 10^{-6}$
Venus	0.5-1.0	0.2-2.5	$0.5 \cdot 1.5 \cdot 10^{-6}$
Sonne/Erde	0.5-2.0	0.0-0.1	$0.4 \cdot 1.5 \cdot 10^{-6}$
Mars	0.5-2.5	0.1-1.0	$3 \cdot 10 \cdot 10^{-6}$
Jupiter	2- 8	0.7-1.0	$20 \cdot 30 \cdot 10^{-6}$
Saturn	2- 11	0.8-1.5	$40 \cdot 100 \cdot 10^{-6}$
Uranus	3- 8	0.7-1.0	$50 \cdot 200 \cdot 10^{-6}$
Neptun	3- 40	1.0-2.0	$500 \cdot 1000 \cdot 10^{-6}$
Pluto	1- 9	0.2-2.2	$200 \cdot 1000 \cdot 10^{-6}$

Die Genauigkeit der Funktionen liegt im 19. und 20. Jahrhundert im Bereich von wenigen Bogensekunden (vgl. Tabelle 6.2). Für entferntere Zeiträume ist allerdings mit schlechteren Ergebnissen zu rechnen. Dies liegt einerseits daran, daß die

zugrundeliegenden Theorien um die Jahrhundertwende aufgestellt wurden und nur an die damals verfügbaren Beobachtungen angepaßt werden konnten. Andererseits sind Störungsterme, die wie T^3 anwachsen, in den Programmen vernachlässigt. Diese Terme sind gegenwärtig sehr klein ($< 0.^{\circ}1$), können aber in historischen Zeiträumen leicht auf das Tausendfache dieses Wertes anwachsen.

6.4 Scheinbare und astrometrische Koordinaten

Schlägt man die Ephemeriden eines Planeten in einem guten Jahrbuch nach, dann findet man dort üblicherweise Angaben darüber, ob es sich bei den abgedruckten Werten um *geometrische*, *scheinbare* oder *astrometrische* Koordinaten handelt. Der Unterschied zwischen diesen drei Varianten ist belanglos, solange man sich mit Genauigkeiten von $1'$ oder schlechter zufrieden gibt. Um die volle Genauigkeit der im letzten Abschnitt angegebenen Programme auch wirklich nutzen zu können, sollte man aber die wesentlichen Korrekturen der Planetenkoordinaten kennen und verstehen.

Während die geometrischen Koordinaten den Punkt im Raum bezeichnen, an dem sich der Planet zu einer bestimmten Zeit t befindet, geben scheinbare und astrometrische Koordinaten die Richtung an, in der der Planet beobachtet wird.

Geometrische Koordinaten werden üblicherweise nur zur Angabe heliozentrischer Planetenpositionen verwendet. Durch die zusätzliche Angabe des Äquinoktiums wird das verwendete Koordinatensystem eindeutig festgelegt. Die Funktionen *MercuryPos...**PlutoPos* liefern *geometrische heliozentrische ekliptikale Koordinaten bezogen auf das Äquinoktium des Datums (d.h. des Berechnungszeitpunkts)*. Ausgenommen hiervon ist natürlich die Funktion *SunPos*, die die entsprechende *geozentrische* Sonnenposition berechnet.

Demgegenüber werden geozentrische Positionen (mit Ausnahme der Entfernung) meist in Form astrometrischer oder scheinbarer Koordinaten angegeben. Der Grund hierfür liegt darin, daß der Planet infolge seiner Bewegung und der Lichtlaufzeit nicht an dem Ort beobachtet wird, an dem er sich zur Beobachtungszeit befindet. Auf diesen Effekt wurde schon im Zusammenhang mit der Berechnung von Kometenbahnen hingewiesen. Da der vom Licht tatsächlich zurückgelegte Weg nicht unmittelbar beobachtbar ist, beschränkt man sich üblicherweise auch in geozentrischen Ephemeriden auf die Angabe der geometrischen Entfernung.

6.4.1 Aberration und Lichtlaufzeit

Astrometrische Koordinaten weisen in die Richtung, aus der ein beobachteter Lichtstrahl vom Planeten zur Erde gelangt ist. Dieser Lichtstrahl verbindet den Erdort zur Beobachtungszeit t mit dem Ort des Planeten zum Zeitpunkt t' der Lichtaussendung. Die Lichtlaufzeit $\tau = t - t'$ ergibt sich näherungsweise aus der geometrischen Entfernung

$$\Delta_0 = |\mathbf{r}_0| \quad \text{mit} \quad \mathbf{r}_0 = \mathbf{r}_\odot(t) + \mathbf{r}_p(t) \quad (6.11)$$

zu $\tau \approx \Delta_0/c$ (vgl. Abb. 6.2 zur Definition der einzelnen Größen). Da sich der Planet in dieser Zeit annähernd geradlinig bewegt, erhält man für die astrometrischen

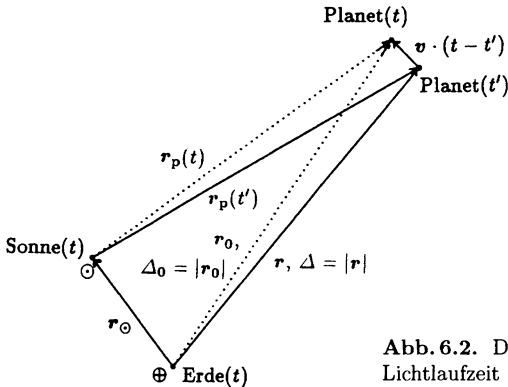


Abb. 6.2. Die Bewegung des Planeten während der Lichtlaufzeit

Koordinaten des Planeten zur Beobachtungszeit t den Ausdruck

$$\begin{aligned} \mathbf{r} &= \mathbf{r}_\odot(t) + \mathbf{r}_p(t') \\ &\approx \mathbf{r}_\odot(t) + \left\{ \mathbf{r}_p(t) - \mathbf{v}_p(t) \cdot \frac{\Delta_0}{c} \right\} \end{aligned} \quad (6.12)$$

Der Unterschied zwischen den *astrometrischen* Koordinaten \mathbf{r} und den geometrischen Koordinaten \mathbf{r}_0 wird auch als *Lichtlaufzeit-Korrektur* bezeichnet. Astrometrische Koordinaten lassen sich direkt in einen Himmelsatlas einzeichnen oder können auf einer Fotoplatte mit bekannten Sternörtern verglichen werden.

Stellt man dagegen den Planeten mit Hilfe der astrometrischen Koordinaten an den Teilkreisen eines parallaktisch montierten Fernrohrs ein, dann zeigt sich eine Abweichung zwischen dem eingestellten und dem beobachteten Ort, die rund $20''$ beträgt. Die Ursache dieses Fehlers ist, daß die astrometrischen Koordinaten die Richtung des auf die Erde einfallenden Lichtstrahls in einem relativ zur Sonne *ruhenden* Bezugssystem angeben. Der Beobachter ist dagegen der täglichen Drehung der Erde und ihrem jährlichen Lauf um die Sonne unterworfen. Aufgrund dieser Bewegung und der endlichen Lichtgeschwindigkeit sieht er eine andere Einfallssrichtung des vom Planeten kommenden Lichtstrahls. Ein vielzitiertes Beispiel zur Veranschaulichung dieses Unterschieds zwischen einem ruhenden und einem bewegten Beobachter kann jeder leicht selbst nachvollziehen: Regentropfen, die senkrecht zur Erde fallen, scheinen für einen Autofahrer je nach seiner Geschwindigkeit mehr oder weniger aus der Richtung zu kommen, in die er sich selbst bewegt.

Die notwendige Korrektur der Koordinaten beim Wechsel vom ruhenden ins bewegte Bezugssystem betrifft neben den Planeten und Kometen auch die Fixsterne. Man spricht deshalb auch von *stellarer Aberration*. Sternpositionen in Katalogen und Himmelsatlanten geben die Örter der Sterne so wieder, wie man sie als im Sonnensystem ruhender Beobachter wahrnehmen würde. Sie entsprechen daher den astrometrischen Koordinaten der Planeten, mit denen sie direkt verglichen werden können. Der Beobachter auf der Erde mißt an seinen Teilkreisen dagegen die scheinbaren Sternpositionen, die periodisch mit dem Lauf des Jahres um die Katalogörter schwanken.

Eine strenge Ableitung der Formeln für die stellare Aberration würde an dieser Stelle etwas zu weit führen, weil es sich dabei eigentlich um ein Phänomen der speziellen Relativitätstheorie handelt. Da die Geschwindigkeit der Erde relativ zur Lichtgeschwindigkeit aber sehr klein ist, führt die folgende halbklassische Argumentation in der hier interessierenden Genauigkeit zum selben Ergebnis.

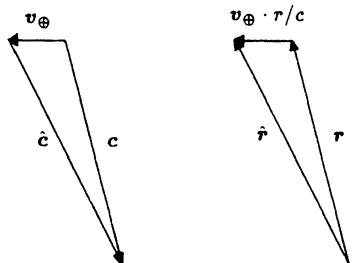


Abb. 6.3. Zur Wirkung derstellaren Aberration

Ist \mathbf{r} der Vektor der astrometrischen Koordinaten des Planeten, dann kann der auf die Erde treffende Lichtstrahl durch den Vektor der Lichtgeschwindigkeit

$$\mathbf{c} = -c \cdot \frac{\mathbf{r}}{|\mathbf{r}|}$$

beschrieben werden (vgl. Abb. 6.3). Den entsprechenden Vektor $\hat{\mathbf{c}}$ im erdgebundenen System des Beobachters erhält man, wenn man davon die Geschwindigkeit \mathbf{v}_\oplus der Erde relativ zur Sonne abzieht:

$$\hat{\mathbf{c}} = \mathbf{c} - \mathbf{v}_\oplus .$$

Der Beobachter sieht den Planeten damit in der Richtung

$$\begin{aligned}\hat{\mathbf{r}} &= -|\mathbf{r}| \cdot \frac{\hat{\mathbf{c}}}{c} = \mathbf{r} + \mathbf{v}_\oplus \cdot \frac{|\mathbf{r}|}{c} \\ &\approx \mathbf{r} + \mathbf{v}_\oplus \cdot \frac{\Delta_0}{c} .\end{aligned}\tag{6.13}$$

Die so bestimmten Koordinaten werden als *scheinbare* Position des Planeten bezeichnet, wenn zusätzlich das verwendete Koordinatensystem genau in Richtung der aktuellen Lage der Erdachse orientiert ist. Die dafür erforderliche Wahl des Äquinoktiums und die Berücksichtigung der Nutation sind das Thema des nächsten Abschnitts.

Zuvor soll aber noch gezeigt werden, wie man den heliozentrischen Geschwindigkeitsvektor der einzelnen Planeten berechnen kann, den man zur Berücksichtigung der verschiedenen Aberrationseffekte benötigt. Da die Geschwindigkeit v jedes einzelnen Planeten klein gegen die Lichtgeschwindigkeit c ist, liegen die durch Lichtlaufzeit und stellare Aberration bedingten Korrekturen selbst nur in der Größenordnung von einer Bogenminute. Entsprechend genügt es, die Geschwindigkeiten der Planeten auf rund zwei Dezimalen genau zu kennen, wenn man die Aberration auf eine Bogensekunde genau bestimmen will. Anstelle aufwendiger Reihenentwicklungen genügt hier also die Berechnung der Geschwindigkeiten auf

der Grundlage genauerer Keplerbahnen. Analog zu der bereits früher vorgestellten Funktion `KepPosition` (vgl. Abschn. 5.2) liefert `KepVelocity` den heliozentrischen Geschwindigkeitsvektor eines Planeten.

```
-----
// KepVelocity: Geschwindigkeit eines Planeten aus Keplerschen Elementen
//   Planet    Identifiziert den Planeten
//   T        Zeit in Julianischen Jahrhunderten seit J2000
//   <return> Heliozentrische Geschwindigkeit [AE/d], Ekliptik und
//           Aequinoktium des Datums
-----
Vec3D KepVelocity (PlanetType Planet, double T)
{
    Vec3D r, v;
    State (Planet, T, r, v);  return v;
}
```

Die in `State` verwendeten Bahnelemente gelten exakt für den Beginn des Jahres 2000, liefern aber auch mehrere Jahre vor und nach diesem Zeitpunkt ausreichend genaue Ergebnisse.

6.4.2 Die Nutation

Scheinbare Koordinaten sind Koordinaten, die für die Beobachtung an einem mit Teikreisen versehenen Fernrohr vorgesehen sind. Da die Montierung des Fernrohrs immer fest in Richtung der Erdachse ausgerichtet ist, muß auch das verwendete Koordinatensystem entsprechend orientiert sein. Dies bedingt zunächst, daß sich scheinbare Koordinaten immer auf das Äquinoktium des Datums beziehen, also auf die jeweils aktuelle Lage von Äquator, Ekliptik und Frühlingspunkt. Neben der Präzession ist dabei aber eine weitere Lagestörung der Erdachse zu berücksichtigen, über die bei der bisherigen Behandlung der Koordinatensysteme noch nicht gesprochen wurde. Während die Präzession die säkulare – d.h. langfristige – Verlagerung der Erdachse bezeichnet, versteht man unter der *Nutation* eine zusätzliche kleine Auslenkung periodischer Natur. Der durch die Nutation verschobene *wahre* Pol der Erdachse dreht sich in 18.6 Jahren einmal um den *mittleren* Pol, dessen Bewegung durch die Präzession beschrieben wird. Die Periode der Nutation wird durch die Umlaufszeit des aufsteigenden Knotens Ω der Mondbahn bestimmt. Die Nutation in Länge ($\Delta\psi$), also die Längendifferenz zwischen *wahrem* und *mittlerem* Frühlingspunkt, hat eine Amplitude von rund $17''$. Gleichzeitig variiert die Ekliptikschiefe um $\Delta\epsilon \approx 9''$ (vgl. Abb. 6.4).

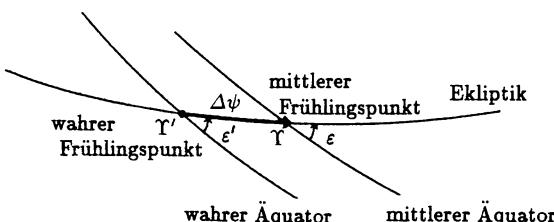


Abb. 6.4. Die Verschiebung von Äquator, Ekliptik und Frühlingspunkt durch die Nutation

Insgesamt setzen sich beide Größen aus rund hundert einzelnen Termen zusammen, von denen für die meisten Zwecke aber nur einige wenige benötigt werden:

$$\begin{aligned}\Delta\psi &= -17''200 \cdot \sin(\Omega) + 0''206 \cdot \sin(2\Omega) + 0''143 \cdot \sin(l') \\ &\quad - 1''319 \cdot \sin(2(F - D + \Omega)) - 0''227 \cdot \sin(2(F + \Omega)) \\ \Delta\varepsilon &= +9''203 \cdot \cos(\Omega) - 0''090 \cdot \cos(2\Omega) \\ &\quad + 0''574 \cdot \cos(2(F - D + \Omega)) + 0''098 \cdot \cos(2(F + \Omega)) .\end{aligned}\quad (6.14)$$

Neben der Länge des aufsteigenden Knotens Ω hängen diese Terme von verschiedenen weiteren Argumenten (F, D, l') ab, die aus den mittleren Längen und Anomalien von Sonne und Mond zusammengesetzt sind (vgl. (8.2) ... (8.5)).

Die wahren äquatorialen Koordinaten $\mathbf{r}' = (x', y', z')$ ergeben sich aus den bisher verwendeten mittleren Koordinaten $\mathbf{r} = (x, y, z)$ durch Multiplikation mit der Nutationsmatrix

$$\mathbf{N} = \mathbf{R}_x(-\varepsilon - \Delta\varepsilon) \mathbf{R}_z(\Delta\psi) \mathbf{R}_x(\varepsilon) , \quad (6.15)$$

mit den Komponenten

$$\begin{aligned}n_{11} &= +\cos(\Delta\psi) \\ n_{21} &= +\cos(\varepsilon') \cdot \sin(\Delta\psi) \\ n_{31} &= +\sin(\varepsilon') \cdot \sin(\Delta\psi) \\ n_{12} &= -\cos(\varepsilon) \cdot \sin(\Delta\psi) \\ n_{22} &= +\cos(\varepsilon) \cdot \cos(\varepsilon') \cdot \cos(\Delta\psi) + \sin(\varepsilon) \cdot \sin(\varepsilon') \\ n_{32} &= +\cos(\varepsilon) \cdot \sin(\varepsilon') \cdot \cos(\Delta\psi) - \sin(\varepsilon) \cdot \cos(\varepsilon') \\ n_{13} &= -\sin(\varepsilon) \cdot \sin(\Delta\psi) \\ n_{23} &= +\sin(\varepsilon) \cdot \cos(\varepsilon') \cdot \cos(\Delta\psi) - \cos(\varepsilon) \cdot \sin(\varepsilon') \\ n_{33} &= +\sin(\varepsilon) \cdot \sin(\varepsilon') \cdot \cos(\Delta\psi) + \cos(\varepsilon) \cdot \cos(\varepsilon')\end{aligned}\quad (6.16)$$

die ihrerseits aus drei elementaren Drehungen zusammengestellt ist (vgl. Abb. 6.4). Hierin ist ε die mittlere Schiefe der Ekliptik aus (2.9) und $\varepsilon' = \varepsilon + \Delta\varepsilon$ die wahre Ekliptikschiefe.

Die Funktion `NutMatrix` erwartet als Eingabe den Zeitpunkt T in julianischen Jahrhunderten seit J2000 und berechnet daraus die Transformation von mittleren zu wahren Koordinaten unter Verwendung der vereinfachten Nutationswinkel.

```
//-----
// NutMatrix: Transformation vom mittleren Aequator und Fruehlingspunkt zum
//            wahren Aequator und Fruehlingspunkt
//      T          Zeit in Julianischen Jahrhunderten seit J2000
//      <return>  Nutationsmatrix
//-----
Mat3D NutMatrix (double T)
{
    double ls, D, F, N;
```

```

double  eps, dpsi, deps;
ls = pi2*Frac(0.993133+ 99.997306*T); // Mittlere Anomalie der Sonne
D  = pi2*Frac(0.827362+1236.853087*T); // Laengendifferenz Sonne-Mond
F  = pi2*Frac(0.259089+1342.227826*T); // Mittleres Argument der Breite
N  = pi2*Frac(0.347346- 5.372447*T); // Laenge des aufsteigenden Knotens
dpsi = ( -17.200*sin(N) - 1.319*sin(2*(F-D+N)) - 0.227*sin(2*(F+N))
        + 0.206*sin(2*N) + 0.143*sin(ls) ) / Arcs;
deps = ( + 9.203*cos(N) + 0.574*cos(2*(F-D+N)) + 0.098*cos(2*(F+N))
        - 0.090*cos(2*N) ) / Arcs;
eps  = 0.4090928-2.2696E-4*T; // Mittlere Schiefe der Ekliptik
return R_x(-eps-deps)*R_z(-dpsi)*R_x(+eps);
}

```

6.5 Das Programm PLANPOS

Das Programm Planpos berechnet die Positionen der Sonne und der neun großen Planeten für einen vorgegebenen Zeitpunkt. Wegen der beschränkten Gültigkeit der Funktion PlutoPos werden die Plutokoordinaten allerdings nur zwischen den Jahren 1890 und 2100 ausgegeben. Neben dem Datum kann das Äquinoktium spezifiziert werden, auf das sich die zu berechnenden Positionen beziehen sollen. Zur Auswahl stehen dabei scheinbare Koordinaten, sowie astrometrische Koordinaten für die derzeit gebräuchlichen Äquinoktien B1950 und J2000. Scheinbare Koordinaten beziehen sich auf das wahre Äquinoktium des Datums und beinhalten Korrekturen für Präzession, Nutation, Aberration und Lichtlaufzeit. Man kann sie zum Beispiel an den Teilkreisen eines parallaktisch montierten Fernrohres einstellen oder ablesen. Die astrometrischen Koordinaten benötigt man, wenn man die Planetenpositionen in eine Himmelskarte des entsprechenden Äquinoktiums eintragen möchte. Sie berücksichtigen Präzession und Lichtlaufzeit.

```

//-----
// Datei: Planpos.cpp
// Zweck: Ephemeriden der grossen Planeten
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
// -----
#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>

#include "APC_Const.h"
#include "APC_Kepler.h"
#include "APC_Math.h"
#include "APC_Planets.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"

```

```

using namespace std;

//-----
// Hauptprogramm
// -----
void main()
{
    // Namen der Himmelskoerper
    const char* Name[] =
        { "Sonne", "Merkur", "Venus", "Erde", "Mars",
          "Jupiter", "Saturn", "Uranus", "Neptun", "Pluto" };

    // Variablen
    PlanetType Planet;
    char      Mode;
    int       iPlanet, last;
    int       year, month, day;
    double   Hour, MJD, T;
    double   dist, fac;
    Vec3D    R_Sun, r_helioc, r_geoc, r_equ;
    Mat3D    P;
    bool     End = false;

    // Titel
    cout << endl
        << "  PLANPOS: geozentrische und heliozentrische Planetenpositionen"
        << endl
        << "           (c) 1999 Oliver Montenbruck, Thomas Pfleger" << endl
        << endl;

    // Kommando-Schleife
    do {
        // Eingabe eines Kommandos
        cout << endl
            << " (J) J2000 astrometrisch      (B) B1950 astrometrisch" << endl
            << " (S) Scheinbare Koordinaten  (E) Ende                  " << endl
            << endl
            << " Kommando ... ";
        cin >> Mode; cin.ignore(81,'\'\n'); Mode = tolower(Mode);
        cout << endl;

        // Aktionen

        // Ende, falls Mode='E'
        if (Mode=='e') { End=true; break; }
        // Wiederhole, falls Kommando ungultig
        if (Mode!='j' && Mode!='b' && Mode!='s') continue;

        // Datum der Epoche
        cout << " Datum (JJJJ MM TT HH.HHHH) ... ";
        cin >> year >> month >> day >> Hour; cin.ignore(81,'\'\n');
        cout << endl << endl << endl;
}

```

```

MJD = Mjd(year,month,day)+Hour/24.0;
T   = ( MJD - MJD_J2000 ) / 36525.0;

// Vorspann
cout << " Datum: " << DateTime(MJD,HHh) << " (ET)"
<< " JD: " << (MJD+2400000.5)
<< setw(18) << "Aequinoktium ";
switch (Mode) {
    case 's': cout << "des Datums" << endl; break;
    case 'j': cout << "J2000" << endl; break;
    case 'b': cout << "B1950" << endl;
}
cout << endl
<< endl
<< "          l           b           r           "
<< "RA        Dec       delta"
<< endl
<< "          o , \\"      o , \\"      AE         "
<< "h  m  s      o , \\"      AE"
<< endl;

// Ekliptikale Koordinaten der Sonne, Aequinoktium des Datums
R_Sun = SunPos(T);

// Schleife ueber die Planeten
if ( (-1.1<T) && (T<1.0) ) last=Pluto; else last=Neptune;
for (iPlanet=Sun; iPlanet<=last; iPlanet++) {

    // Heliozentrische ekliptikale Koordinaten des Planeten;
    // Aequinoktium des Datums
    Planet = (PlanetType) iPlanet;
    r_helioc = PertPosition(Planet, T);
    // Geozentrische ekliptikale Koordinaten (Aequinoktium des Datums)
    r_geoc = r_helioc + R_Sun;
    // Korrektur (1. Ordnung) fuer die Lichtlaufzeit/Aberration
    dist = Norm(r_geoc);
    fac = dist/c_light;
    if (Mode=='s')
        r_geoc -= fac*(KepVelocity(Planet,T)-KepVelocity(Earth,T));
    else
        r_geoc -= fac*KepVelocity(Planet,T);
    // Praezession und aequatoriale Koordinaten
    switch (Mode) {
        case 's': r_equ = NutMatrix(T) * Ecl2EquMatrix(T) * r_geoc;
                    break;
        case 'j': P      = PrecMatrix_Ecl(T,T_J2000);
                   r_helioc = P * r_helioc;
                   r_geoc  = P * r_geoc;
                   r_equ   = Ecl2EquMatrix(T_J2000) * r_geoc;
                    break;
        case 'b': P      = PrecMatrix_Ecl(T,T_B1950);
                   r_helioc = P * r_helioc;
                   r_geoc  = P * r_geoc;
                   r_equ   = Ecl2EquMatrix(T_B1950) * r_geoc;
    }
}

```

```

// Ausgabe
cout << " " << setw(8) << left << Name[iPlanet] << right;
cout << fixed << setprecision(1)
    << setw(11) << Angle(Deg*r_helioc[phi],DMMSSs) << " "
    << setw(11) << Angle(Deg*r_helioc[theta],DMMSSs);
if (Planet<=Earth)
    cout << setprecision(6) << setw(11) << r_helioc[r];
else
    cout << setprecision(5) << setw(10) << r_helioc[r] << " ";
cout << setprecision(2) << setw(13) << Angle(Deg*r_equ[phi]/15.0,DMMSSs)
    << " " << showpos
    << setprecision(1) << setw(11) << Angle(Deg*r_equ[theta],DMMSSs)
    << noshowpos;
if (Planet<=Earth)
    cout << setprecision(6) << setw(11) << dist;
else
    cout << setprecision(5) << setw(10) << dist << " ";
cout << endl;
};

// Nachspann
cout << endl
    << " l,b,r: heliozentrisch ekliptikal (geometrisch) " << endl
    << " RA,Dec: geozentrisch aequatorial ";
if (Mode=='s')
    cout << " (scheinbar)" << endl;
else
    cout << "(astrometrisch)" << endl;
cout << " delta: geozentrische Entfernung (geometrisch)" << endl
    << endl;
}
while (!End);
}

```

Planpos fragt zu Beginn, auf welches Äquinoktium sich die berechneten Positionen beziehen sollen. Mögliche Eingaben sind S (scheinbare Koordinaten, Äquinoktium des Datums), J (J2000, astrometrisch) und B (B1950, astrometrisch). Diese Optionsabfrage erscheint außer zu Beginn auch nach jedem Rechenlauf. Hierbei ist es auch möglich, den Programmablauf zu beenden (E).

Im folgenden Beispiel wählen wir zunächst scheinbare Koordinaten. Nach der entsprechenden Abfrage ist der genaue Zeitpunkt anzugeben, für den die Planetenstellungen berechnet werden sollen. Man beachte, daß es sich dabei um eine Zeitangabe in Ephemeridenzeit (ET) handelt! Im Beispiel wählen wir den 1. Januar 1989 um 0^h ET. Alle Eingaben sind durch kursive Schrift hervorgehoben. Planpos gibt nun die gewünschten Koordinaten aus.

PLANPOS: geozentrische und heliozentrische Planetenpositionen
(c) 1999 Oliver Montenbruck, Thomas Pfleger

(J) J2000 astrometrisch (B) B1950 astrometrisch
(S) Scheinbare Koordinaten (E) Ende

Kommando ... s

Datum (JJJJ MM TT HH.HHH) ... 1989 01 01 00.0

Datum: 1989/01/01 00.0 (ET) JD: 2447527.5 Aequinoktium des Datums

	l o , "	b o , "	r AE	RA h m s	Dec o , "	delta AE
Sonne	0 00 00.0	0 00 00.0	0.000000	18 45 53.66	-23 01 25.6	0.983309
Merkur	347 56 35.7	- 6 05 21.5	0.370100	19 59 16.60	-22 34 12.1	1.175632
Venus	226 05 15.0	1 43 26.9	0.723666	17 07 15.21	-22 03 57.5	1.522289
Erde	100 33 09.4	0 00 00.2	0.983309	0 00 00.00	+ 0 00 00.0	0.000000
Mars	60 32 51.9	0 21 19.3	1.49745	1 13 47.44	+ 8 24 05.2	0.97649
Jupiter	64 30 07.8	- 0 45 53.2	5.03195	3 38 35.14	+18 33 07.5	4.27637
Saturn	275 06 53.5	0 47 14.5	10.04354	18 24 14.50	-22 36 28.2	11.02274
Uranus	271 18 42.6	- 0 13 47.8	19.31483	18 07 39.22	-23 39 00.8	20.28599
Neptun	279 54 35.7	0 55 55.8	30.21917	18 42 54.09	-22 10 16.6	31.20230
Pluto	222 55 15.8	15 52 34.4	29.65861	15 06 35.06	- 1 16 18.1	30.17673

l,b,r: heliozentrisch ekliptikal (geometrisch)

RA,Dec: geozentrisch aequatorial (scheinbar)

delta: geozentrische Entfernung (geometrisch)

Zu beachten ist noch, daß die heliozentrischen ekliptikalnen Koordinaten der Sonne ebenso wie die geozentrischen äquatorialen Koordinaten der Erde stets zu Null gesetzt werden.

Wir wollen nun zum Vergleich die Planetenpositionen für denselben Zeitpunkt, aber jetzt bezogen auf das Äquinoktium J2000 berechnen lassen. Die kleinen Unterschiede, die sich dabei ergeben, sind durch die Präzession für etwa 11 Jahre bedingt.

Kommando ... j

Datum (JJJJ MM TT HH.HHH) ... 1989 01 01 00.0

Datum: 1989/01/01 00.0 (ET) JD: 2447527.5 Aequinoktium J2000

	l o , "	b o , "	r AE	RA h m s	Dec o , "	delta AE
Sonne	0 00 00.0	0 00 00.0	0.000000	18 46 34.57	-23 00 32.4	0.983309
Merkur	348 05 49.4	- 6 05 22.1	0.370100	19 59 56.54	-22 32 12.4	1.175632
Venus	226 14 28.3	1 43 22.9	0.723666	17 07 55.78	-22 04 40.7	1.522289
Erde	100 42 22.5	0 00 05.2	0.983309	0 00 00.00	+ 0 00 00.0	0.000000
Mars	60 42 05.0	0 21 24.1	1.49745	1 14 21.40	+ 8 27 27.7	0.97649
Jupiter	64 39 21.0	- 0 45 48.3	5.03195	3 39 11.61	+18 35 03.6	4.27637
Saturn	275 16 06.7	0 47 09.5	10.04354	18 24 55.40	-22 35 56.1	11.02274
Uranus	271 27 55.8	- 0 13 52.9	19.31483	18 08 20.48	-23 38 44.9	20.28599
Neptun	280 03 48.9	0 55 50.8	30.21917	18 43 34.77	-22 09 26.4	31.20230
Pluto	223 04 29.9	15 52 30.6	29.65861	15 07 09.55	- 1 18 40.5	30.17673

l,b,r: heliozentrisch ekliptikal (geometrisch)

RA,Dec: geozentrisch aequatorial (astrometrisch)

delta: geozentrische Entfernung (geometrisch)

Man kann erkennen, daß die Unterschiede zwar klein, aber doch zu beachten sind. Beim Vergleich mit Jahrbüchern ist deshalb etwas Aufmerksamkeit erforderlich. Zum Teil werden nur die Positionen der Planeten Merkur bis Neptun als scheinbare Koordinaten gegeben, während man für Plutoephemeriden die astrometrischen Koordinaten benutzt. Der Grund hierfür ist, daß man bei der Beobachtung dieses lichtschwachen Planeten auf den Vergleich mit einem Atlas oder einer Aufsuchkarte angewiesen ist, um den Planeten auch identifizieren zu können.

- (J) J2000 astrometrisch (B) B1950 astrometrisch
(S) Scheinbare Koordinaten (E) Ende

Eingabe: *E*

Auf die nächste Abfrage hin wird das Programm durch die Eingabe *E* beendet.

7. Physische Planetenephemeriden

Im vorangegangenen Kapitel haben wir das Rüstzeug für die Berechnung genauer Positionen der großen Planeten bereitgestellt. Wer die Planeten, die Sonne oder den Mond häufiger beobachtet, wird sich auch für diejenigen Angaben interessieren, die den Anblick eines Himmelskörpers im Fernrohr beschreiben. Die visuelle Helligkeit und der scheinbare Winkeldurchmesser sind dabei Daten von grundsätzlicher Bedeutung. Wer Oberflächenmerkmale oder atmosphärische Strukturen beobachten und kartieren will, muß zusätzlich wissen, wie die Rotationsachse des Planeten und damit das planetographische Koordinatensystem relativ zur Blickrichtung orientiert ist und welche Beleuchtungsverhältnisse vorliegen. Wir wollen erläutern, wie diese Angaben berechnet werden können. Dabei streben wir eine Genauigkeit an, wie sie für praktische Beobachtungen in nahezu allen Fällen ausreicht.

Die bei der Berechnung benötigten Daten für Größe, Form und Rotation der Planeten sind durchgängig dem im Literaturverzeichnis aufgeführten „Report of the IAU/IAG/COSPAR Working Group on Cartographic Coordinates and Rotational Elements“ von 1994 entnommen. Die damit bestimmten physischen Ephemeriden sind konsistent mit den zur Zeit im Astronomical Almanac veröffentlichten Werten.

Da die physischen Ephemeriden von der Stellung von Sonne, Erde und Planet relativ zueinander abhängen, benötigen wir für ihre Berechnung die heliozentrischen Koordinaten der Erde und der Planeten. Wir greifen dazu auf die Funktion `PertPosition` zurück, die bereits in Kap. 6 vorgestellt wurde und genaue Planetenpositionen auch über längere Zeiträume liefert.

7.1 Rotation

Ähnlich wie die Drehung der Erde durch die Richtung des Himmelsnordpols und die Sternzeit festgelegt ist, läßt sich auch die Rotation der übrigen Planeten durch die Lage des jeweiligen Nordpols und eines Drehwinkels vollständig beschreiben. Gemäß einer Übereinkunft der Internationalen Astronomischen Vereinigung (IAU) wird dabei als Nordpol derjenige Pol eines Planeten, Kleinplaneten oder Mondes bezeichnet, der auf der Nordseite der sogenannten *unveränderlichen Ebene* des Sonnensystems steht. Vereinfacht kann man sich diese Ebene, die senkrecht auf dem Gesamtdrehimpulsvektor des Sonnensystems steht, als die mittlere Bahn-ebene aller Planeten vorstellen. Da die Bahnen der großen Planeten nur wenig gegen die Erdbahnebene geneigt sind, ist die unveränderliche Ebene näherungsweise mit der Ekliptik identisch.

Tabelle 7.1. Orientierung der Rotationsachsen der Sonne und der Planeten (IAU 1994)

	α_0 (J2000)	δ_0 (J2000)	
Sonne	286°13	+63°87	
Merkur	281°01 –0°033T	+61°45 –0°005T	
Venus	272°76	+67°16	
Erde	0°00 –0°641T	+90°00 –0°557T	
Mars	317°681–0°108T	+52°886–0°061T	
Jupiter	268°05 –0°009T	+64°49 +0°003T	
Saturn	40°589–0°036T	+83°537–0°004T	
Uranus	257°311	–15°175	
Neptun	299°36 +0°70 sin(N)	+43°46 –0°51 cos(N)	$N = 357°85 + 52°316T$
Pluto	313°02	+9°09	

Nach der genannten Definition rotieren nicht alle Planeten wie die Erde. Zu den Ausnahmen gehören Venus, Uranus und Pluto, die sich beim Blick auf den Nordpol im Uhrzeigersinn drehen. Im Gegensatz zur sogenannten *direkten Rotation* der meisten Planeten wird die Rotation dieser Planeten als *retrograd* bezeichnet.

Zusätzlich zur Berechnung des Rotationszustandes der großen Planeten wollen wir die entsprechenden Angaben für die Sonne ermitteln. Wenn im folgenden von Planeten die Rede ist, so wollen wir daran denken, daß die beschriebene Vorgehensweise zur Berechnung des Rotationszustands auf den Fall der Sonne übertragen werden kann. Besonderheiten behandeln wir in einem eigenen Abschnitt.

Eine Übersicht über die Orientierung der Rotationsachsen der Sonne und der Planeten ist in Tabelle 7.1 wiedergegeben. Angegeben sind dabei die Rektaszension α_0 und die Deklination δ_0 des Nordpols bezogen auf das Äquinoktium J2000. Das Argument

$$T = (\text{JD} - 2451545)/36525 \quad (7.1)$$

bezeichnet wie gewohnt die Anzahl julianischer Jahrhunderte seit der Epoche J2000.

7.1.1 Der Positionswinkel der Achse

Aus der Sicht eines Beobachters beschreibt man die Lage der Rotationsachse eines Planeten zweckmäßigerweise durch die Angabe des Positionswinkels der Achse und der Breite der Erde in Bezug auf den Äquator des Planeten (siehe Abb. 7.1). Unter dem Positionswinkel ϑ der Planetenachse versteht man dabei den Winkel zwischen der Projektion der Rotationsachse an die Himmelssphäre und der Nordrichtung des Beobachters. Der Positionswinkel wird von Nord ($\vartheta = 0^\circ$) über Ost ($\vartheta = 90^\circ$) gemessen.

Zur Berechnung des Positionswinkels der Planetenachse gehen wir von den geozentrischen äquatorialen Koordinaten $\mathbf{r} = (x, y, z)$ des Planeten aus. Weiterhin

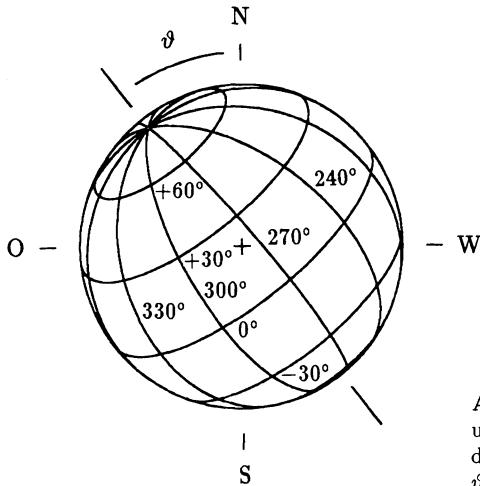


Abb. 7.1. Positionswinkel der Rotationsachse und planetographische Koordinaten (Anblick des Planeten Mars am 1. Okt. 1993, 0^h ET; $\vartheta = 37^\circ 9'$, $\lambda_\oplus = 276^\circ 1'$, $\varphi_\oplus = 20^\circ 4'$)

sei der Richtungsvektor

$$\mathbf{d} = \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} = \begin{pmatrix} \cos \alpha_0 \cos \delta_0 \\ \sin \alpha_0 \cos \delta_0 \\ \sin \delta_0 \end{pmatrix} \quad (7.2)$$

gegeben, der vom Planetenmittelpunkt in Richtung der Achse zeigt. Wir konstruieren nun unter Zuhilfenahme des Vektors \mathbf{r} die drei aufeinander senkrecht stehenden Einheitsvektoren $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$, so daß \mathbf{e}_1 in Richtung zum Planeten, \mathbf{e}_2 in Richtung aufsteigender Rektaszension (nach Osten) und $\mathbf{e}_3 = \mathbf{e}_1 \times \mathbf{e}_2$ in Richtung aufsteigender Deklination (nach Norden) zeigt (vgl. Abb. 7.2). Mit dieser Festlegung gilt

$$\mathbf{e}_1 = \frac{1}{r} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \mathbf{e}_2 = \frac{1}{\varrho} \begin{pmatrix} -y \\ +x \\ 0 \end{pmatrix} \quad \mathbf{e}_3 = \frac{1}{r\varrho} \begin{pmatrix} -xz \\ -yz \\ x^2 + y^2 \end{pmatrix}, \quad (7.3)$$

wobei

$$r = \sqrt{x^2 + y^2 + z^2} \quad \text{und} \quad \varrho = \sqrt{x^2 + y^2}$$

den Betrag des Radiusvektors und seiner Projektion auf die Äquatorebene bezeichnen. Der Positionswinkel ϑ , unter dem wir den Richtungsvektor \mathbf{d} sehen, wird in der durch \mathbf{e}_2 und \mathbf{e}_3 aufgespannten Ebene gemessen, die senkrecht zur Blickrichtung steht. Durch Projektion des Richtungsvektors \mathbf{d} auf \mathbf{e}_2 und \mathbf{e}_3 erhalten wir folgende Beziehungen zur Bestimmung des Positionswinkels ϑ :

$$\begin{aligned} \cos \vartheta &= \mathbf{e}_3 \cdot \mathbf{d} = ((-xz)d_x + (-yz)d_y + (x^2 + y^2)d_z)/(r\varrho) \\ \sin \vartheta &= \mathbf{e}_2 \cdot \mathbf{d} = ((-y)d_x + (x)d_y)/\varrho \end{aligned} \quad (7.4)$$

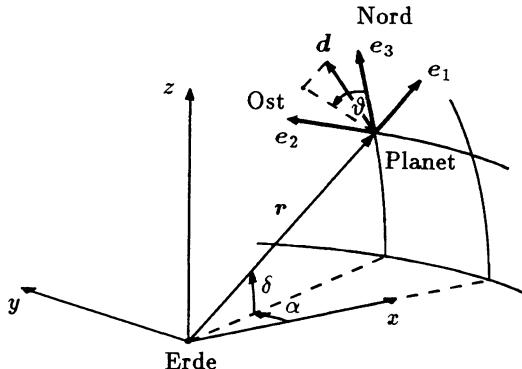


Abb. 7.2. Zur Berechnung des Positionswinkels

Die Funktion `PosAng` faßt diese Formeln zur Berechnung des Positionswinkels zusammen:

```
-----
// PosAng: Berechnet den Positionswinkel des Richtungsvektors d bezogen auf
//          den Ortsvektor r
// r          Ort des beobachteten Himmelskoerpers
// d          Richtung, deren Positionswinkel zu berechnen ist
// <return>   Positionswinkel in [rad]; Bereich [0, 2*pi]
// Beachte: Koordinatensystem und Epoche fuer r und d muessen uebereinstimmen
-----
double PosAng (const Vec3D& r, const Vec3D& d)
{
    // Variablen
    Vec3D e_1, e_2, e_3;
    double c, s, phi;

    // Einheitsvektoren in radialer, oestlicher und noerdlicher Richtung
    e_1 = r / Norm(r);
    e_2 = Cross ( Vec3D(0, 0, 1), r );
    e_2 = e_2 / Norm(e_2);
    e_3 = Cross ( e_1, e_2 );

    // Positionswinkel
    c = Dot(d, e_3);
    s = Dot(d, e_2);
    phi = atan2(s, c);

    return Modulo(phi, 2.0*pi);
}
```

Die Funktion `PosAng` kann auch dazu verwendet werden, den Positionswinkel der Sonne zu berechnen, indem man anstelle des Richtungsvektors d die planetozentrischen Koordinaten der Sonne einsetzt. Hierauf werden wir im Abschn. 7.2 in Zusammenhang mit der Beleuchtungsgeometrie der Planeten noch ausführlich zu sprechen kommen.

7.1.2 Planetographische Koordinaten

Analog zu den geographischen Koordinaten legt man auch für die Planeten ein körperfestes sphärisches Koordinatensystem für Positionsangaben auf der Oberfläche fest. Als Bezugsebene dient dabei der Äquator des jeweiligen Himmelskörpers. Da die Mehrzahl der Planeten unseres Sonnensystems infolge ihrer Rotation abgeplattet ist, unterscheidet man zwischen zwei Definitionen der Breite:

- Die *planetozentrische* Breite φ' bezeichnet den Winkel zwischen dem Richtungsvektor eines Punktes und der Äquatorebene des Planeten. Sie entspricht damit der Deklination im geozentrischen äquatorialen Koordinaten- system.
- Die *planetographische* Breite φ weicht von der planetozentrischen Breite φ' ab, wenn der Planet abgeplattet ist. Bildet man von einem gegebenen Punkt aus das Lot auf die Oberfläche eines Planeten, dann bezeichnet die planetographische Breite den mit der Äquatorebene eingeschlossenen Winkel. Sie entspricht damit der im Falle der Erde verwendeten geographischen Breite (siehe Abb. 9.5).

Der Zusammenhang zwischen der geozentrischen Breite φ' und der geographischen Breite φ ist in Abschn. 9.3 beschrieben. Wir brauchen den Erdradius R_{\oplus} nur durch den Äquatorradius $R_{\text{Äq}}$ des Planeten zu ersetzen, wobei die Abplattung des Planetenellipsoids als

$$f = \frac{R_{\text{Äq}} - R_{\text{Pol}}}{R_{\text{Äq}}} \quad (7.5)$$

definiert ist. Die Umrechnung zwischen planetozentrischer Breite φ' und planetographischer Breite φ kann demnach mit der Gleichung

$$\tan \varphi' = (1 - f)^2 \tan \varphi \quad (7.6)$$

erfolgen.

Für die Berechnung der physischen Ephemeriden stellen wir die Werte für die Äquatorradien und die Abplattung der großen Planeten zweckmäßigerweise mit der Funktion `Shape` bereit, um eine Entkopplung der Daten von den Berechnungsroutinen zu erreichen.

```
//-----
// Shape: Liefert den Äquatorradius und die Abplattung der Planeten
//   Planet      Identifiziert den Planeten (s. APC_Planets.h)
//   R_equ      Äquatorradius in [km]
//   f          Geometrische Abplattung
//-----
void Shape (PlanetType Planet, double& R_equ, double& f)
{
    switch (Planet) {
        case Sun:      R_equ = 696000.0;  f = 0.0;           break;
        case Mercury:  R_equ = 2439.0;   f = 0.0;           break;
        case Venus:    R_equ = 6051.0;   f = 0.0;           break;
```

```

case Earth:    R_equ =  6378.14; f = 0.00335281; break;
case Mars:     R_equ =  3393.4;  f = 0.0051865;  break;
case Jupiter:  R_equ = 71398.0;  f = 0.0648088; break;
case Saturn:   R_equ = 60000.0; f = 0.1076209; break;
case Uranus:   R_equ = 25400.0; f = 0.030;      break;
case Neptune:  R_equ = 24300.0; f = 0.0259;     break;
case Pluto:    R_equ = 1500.0;  f = 0.0;
}
}

```

Dabei wird der bereits früher definierte Aufzählungstyp

```
enum PlanetType { Sun, Mercury, Venus, Earth, Mars,
                  Jupiter, Saturn, Uranus, Neptune, Pluto },
```

zur Kennzeichnung der einzelnen Planeten verwendet.

Ähnlich wie bei der Festlegung der Breite eines Punktes wird auch bei der Längenzählung zwischen der planetozentrischen und der planetographischen Länge unterschieden:

- Die *planetozentrische* Länge λ' wird unabhängig von der Rotationsrichtung positiv in Richtung Osten gezählt.
- Die *planetographische* Länge λ wird entgegen der Rotationsrichtung des Planeten gezählt. Gemäß dieser Definition nimmt die Länge des sogenannten *Zentralmeridians*, der durch die Mitte der von der Erde aus gesehenen Planetenscheibe verläuft, mit der Zeit fortlaufend zu.

Für die retrograd rotierenden Planeten Venus, Uranus und Pluto sind beide Längendefinitionen identisch ($\lambda = \lambda'$). Dagegen unterscheiden sich die planetozentrische Länge und die planetographische Länge bei den Planeten Merkur, Mars, Jupiter, Saturn und Neptun, die wie die Erde von West nach Ost rotieren, durch das Vorzeichen ($\lambda = -\lambda'$). Im Fall der Sonne ist allerdings zu beachten, daß die sogenannte heliographische Länge in Rotationsrichtung zunimmt und damit identisch mit der heliozentrischen Länge ist, obwohl die Sonne eine direkte Rotation aufweist.

Die Festlegung einer Bezugsrichtung zur Längenzählung orientiert sich entweder an markanten Strukturen der Oberfläche oder wird bei deren Fehlen in geeigneter Weise definiert. Bei den Riesenplaneten Jupiter, Saturn, Uranus und Neptun, die nicht wie starre Körper mit einer festen Oberfläche rotieren, hat man dazu Rotationssysteme mit konstanter Winkelgeschwindigkeit eingeführt. Für die Jupiteratmosphäre, die am Äquator mit einer Periode von $9^{\text{h}}30^{\text{m}}$, in höheren Breiten aber mit $9^{\text{h}}56^{\text{m}}$ rotiert, bezeichnet das System I die äquatornahen Bereiche ($|\varphi| \leq 9^{\circ}$) während das System II für die gemäßigten und höheren Breiten gilt. Wie radioastronomische Untersuchungen der äußeren Planeten zeigen, rotieren die Magnetfelder dieser Planeten mit Perioden, die nicht mit den Rotationszeiten der optisch fixierten Rotationssysteme I oder II übereinstimmen. Die Ursache hierfür liegt in der Entstehung der Radiostrahlung im Planeteninneren, das weitgehend starr und unabhängig von den sichtbaren Atmosphärenschichten rotiert. Daher

wurde zusätzlich das System III eingeführt, das die Rotation in Bezug auf die Radioemissionen beschreibt. Für Uranus und Neptun wird heute nur noch das Rotationssystem III benutzt. Für Saturn geben wir in Ergänzung zum System III auch die Werte des Systems I an, um die Vergleichsmöglichkeit mit alten Aufzeichnungen nicht zu verlieren.

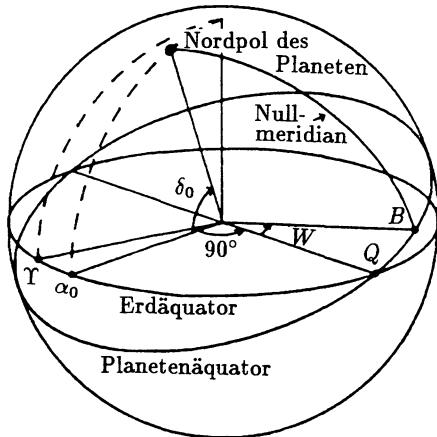


Abb. 7.3. Orientierung des Nullmeridians

Um die Lage des Nullmeridians eines Planeten zu beschreiben, verwenden wir den Winkel W . Er wird

- ausgehend vom Schnittpunkt des Erdäquators (J2000) und des Planetenäquators (Punkt Q)
- längs des Planetenäquators
- in Rotationsrichtung anwachsend
- bis zum Schnittpunkt zwischen dem Nullmeridian und dem Planetenäquator (Punkt B)

gemessen. Dabei ist Punkt Q als derjenige der beiden Schnittpunkte von Erd- und Planetenäquator definiert, in dem ein am Äquator des Planeten mitrotierender Punkt die Ebene des Erdäquators von Süden nach Norden hin durchstößt (vgl. Abb. 7.3). Der Winkel W nimmt linear mit der Zeit zu und kann aus den Angaben der Tabelle 7.2 berechnet werden. Das Argument

$$d = \text{JD} - 2451545 \quad (7.7)$$

bezeichnet darin die Anzahl der seit der Standardepoche J2000 verstrichenen Tage.

Die numerischen Werte der Parameter α_0 , δ_0 und W zur Beschreibung der Lage von Rotationsachse und Nullmeridian sind in der Funktion `Orient` zusammengefaßt. Auf diese Weise lassen sich zukünftige Änderungen in den Rotationsparametern besonders einfach berücksichtigen. Neben der Zeit und dem Namen

Tabelle 7.2. Orientierung des Nullmeridians der Sonne und der Planeten (IAU 1994)

		W (J2000)		
Sonne		84°182	+	14°1844000d
Merkur		329°68	+	6°1385025d
Venus		160°20	-	1°4813688d
Erde		190°16	+	360°9856235d
Mars		176°901	+	350°8919830d
Jupiter	System I	67°1	+	877°900d
	System II	43°3	+	870°270d
	System III	284°695	+	870°536d
Saturn	System I	227°2037	+	844°300d
	System III	38°90	+	810°7939024d
Uranus	System III	203°81	-	501°1600928d
Neptun	System III	253°18	+	536°3128492d - 0°48 sin(N) mit $N = 357°85 + 52°316T$
Pluto		236°77	-	56°3623195d

des jeweiligen Planeten verlangt `Orient` noch die Angabe des Rotationssystems. Hierzu dient die Variable `System` vom Typ

```
enum SystemType { Sys_I, Sys_II, Sys_III }; // Rotationssysteme (Jup./Sat.)
```

Je nach Wert dieses Parameters liefert `Orient` für Jupiter und Saturn in `W` die Position des Nullmeridians für äquatornahe Gebiete, höhere Breiten oder Radiostrahlung zurück. Der Parameter `Sense` vom Typ

```
enum RotationType { Direct, Retrograde }; // Rotationsrichtung
```

gibt darüberhinaus an, ob der betreffende Planet direkt oder retrograd rotiert.

```
//-----
// Orient: Berechnet die Orientierung des planetozentrischen Bezugssystems und
//          liefert die Rotationsrichtung
//  Planet   Identifiziert den Planeten (s. APC_Planets.h)
//  System   Rotationssystem (nur bei Jupiter und Saturn verwendet)
//  T        Zeit in Julianischen Jahrhunderten seit J2000
//  E        Transformationsmatrix vom geozentrischen, auf den mittleren
//          Aequator und das Aequinoktium J2000 bezogenen Koordinatensystem
//          zum koerperfesten aequatorialen Bezugssystem
//  Sense    Rotationsrichtung (direkt oder retrograd)
//-----
void Orient ( PlanetType Planet, SystemType System, double T,
              Mat3D& E, RotationType& Sense )
{
    double N, RA, Dec, W;
    double d = 36525.0*T;
    // Rektaszension und Deklination der Rotationsachse bezogen auf
    // Aequator und Ekliptik zur Epoche J2000; Orientierung des
    // Nullmeridians
```

```

switch (Planet) {
    case Sun:      RA = 286.13;
                    Dec = 63.87;
                    W = 84.182 + 14.1844000*d; break;
    case Mercury: RA = 281.01 - 0.033*T;
                    Dec = 61.45 - 0.005*T;
                    W = 329.68 + 6.1385025*d; break;
    case Venus:   RA = 272.76;
                    Dec = 67.16;
                    W = 160.20 - 1.4813688*d; break;
    case Earth:   RA = 0.00 - 0.641*T;
                    Dec = 90.00 - 0.557*T;
                    W = 190.16 + 360.9856235*d; break;
    case Mars:    RA = 317.681 - 0.108*T;
                    Dec = 52.886 - 0.061*T;
                    W = 176.901 + 350.8919830*d; break;
    case Jupiter: RA = 268.05 - 0.009*T;
                    Dec = 64.49 + 0.003*T;
                    switch (System) {
                        case Sys_I : W = 67.10 + 877.900*d; break;
                        case Sys_II : W = 43.30 + 870.270*d; break;
                        case Sys_III : W = 284.695 + 870.536*d; break;
                    }
                    break;
    case Saturn:  RA = 40.589 - 0.036*T;
                    Dec = 83.537 - 0.004*T;
                    switch (System) {
                        case Sys_I :
                            case Sys_II : W = 227.2037 + 844.3000000*d; break;
                            case Sys_III : W = 38.90 + 810.7939024*d; break;
                        }
                    break;
    case Uranus:  RA = 257.311;
                    Dec = -15.175;
                    W = 203.81 - 501.1600928*d; break; // System III
    case Neptune: N = Rad*(357.85+52.316*T);
                    RA = 299.36 + 0.70*sin(N);
                    Dec = 43.46 - 0.51*cos(N);
                    W = 253.18 + 536.3128492*d - 0.48*sin(N); break;
    case Pluto:   RA = 313.02;
                    Dec = 9.09;
                    W = 236.77 - 56.3623195*d;
}
RA*=Rad; Dec*=Rad; W=Rad*Modulo(W,360.0);

// Transformation vom mittleren Erdaequator und Fruehlingspunkt J2000
// zum koerperfesten Bezugssystem von Aequator und Nullmeridian
E = R_z(W) * R_x(pi/2.0-Dec) * R_z(pi/2.0+RA);
// Rotationsrichtung
if ( Planet==Venus || Planet==Uranus || Planet==Pluto )
    Sense = Retrograde;
else
    Sense = Direct;
}

```

Kennt man die Lage der Rotationsachse und des Nullmeridians eines Planeten, so lassen sich daraus die planetozentrischen und planetographischen Koordinaten von Sonne und Erde bestimmen, die für die Beobachtung eines Planeten von besonderem Interesse sind. Diese Größen beschreiben zum Beispiel, wie weit der Nordpol des Planeten in Richtung Erde geneigt ist, welcher Punkt der Plane-tenoberfläche in der Mitte des scheinbaren Planetenscheibchens liegt oder welcher Punkt senkrecht von der Sonne beschienen wird.

Zur Berechnung der planetozentrischen Länge und Breite der Erde oder der Sonne verwenden wir ein körperfestes Koordinatensystem, das mit dem Planeten rotiert und von den drei aufeinander senkrecht stehenden Einheitsvektoren (e_1, e_2, e_3) aufgespannt wird. Der Vektor e_3 ist dabei parallel zur Rotationsachse orientiert, während e_1 vom Mittelpunkt des Planeten zum Schnittpunkt (B) von Äquator und Nullmeridian zeigt. Ausgeschrieben lauten die Komponenten dieser Vektoren in äquatorialen Koordinaten

$$\begin{aligned} \mathbf{e}_1 &= \begin{pmatrix} -\cos W \sin \alpha_0 - \sin W \sin \delta_0 \cos \alpha_0 \\ +\cos W \cos \alpha_0 - \sin W \sin \delta_0 \sin \alpha_0 \\ +\sin W \cos \delta_0 \end{pmatrix} \\ \mathbf{e}_2 &= \begin{pmatrix} +\sin W \sin \alpha_0 - \cos W \sin \delta_0 \cos \alpha_0 \\ -\sin W \cos \alpha_0 - \cos W \sin \delta_0 \sin \alpha_0 \\ +\cos W \cos \delta_0 \end{pmatrix} \\ \mathbf{e}_3 &= \begin{pmatrix} +\cos \delta_0 \cos \alpha_0 \\ +\cos \delta_0 \sin \alpha_0 \\ +\sin \delta_0 \end{pmatrix}. \end{aligned} \quad (7.8)$$

Man erhält diese Beziehungen analog zur bekannten Darstellung der Gaußschen Vektoren \mathbf{P} , \mathbf{Q} und \mathbf{R} (vgl. Abschn. 4.5), wenn man die Bahnelemente i , Ω und ω durch den Winkel $90^\circ - \delta_0$ zwischen Erd- und Planetenäquator, die Rektaszension $\alpha_0 + 90^\circ$ der Schnittlinie von Erd- und Planetenäquator und den Winkel W zwischen dieser Schnittlinie und dem Nullmeridian ersetzt. Entsprechend kann die Matrix

$$\mathbf{E} = \begin{pmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \\ \mathbf{e}_3^T \end{pmatrix} = \mathbf{R}_z(W) \mathbf{R}_x(\pi/2 - \delta) \mathbf{R}_z(\pi/2 + \alpha) \quad (7.9)$$

zur Umrechnung von äquatorialen Koordinaten in das planetenfeste System als Folge von drei elementaren Drehmatrizen dargestellt werden. Diese Darstellung ist besonders kompakt und wird deshalb in Orient anstelle der expliziten Beziehungen (7.8) verwendet.

Bezeichnet man mit $\mathbf{r} = (x, y, z)$ die planetozentrischen Koordinaten der Erde, dann erhält man durch Projektion von \mathbf{r} auf die Richtungsvektoren e_1, e_2 und e_3 die kartesischen planetozentrischen Koordinaten

$$\begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix} = \mathbf{E}\mathbf{r} = \begin{pmatrix} r \cos \varphi' \cos \lambda' \\ r \cos \varphi' \sin \lambda' \\ r \sin \varphi' \end{pmatrix} \quad (7.10)$$

der Erde, aus denen sich über

$$\begin{aligned}
 \tan \varphi' &= \frac{s_z}{\sqrt{s_x^2 + s_y^2}} \\
 \tan \lambda' &= \frac{s_y}{s_x} \\
 \tan \varphi &= \frac{\tan \varphi'}{(1-f)^2} \\
 \lambda &= \begin{cases} +\lambda' & (\text{retrograde Rotation}) \\ -\lambda' & (\text{direkte Rotation}) \end{cases}
 \end{aligned} \tag{7.11}$$

die zugehörigen Werte für die planetozentrische und planetographische Länge und Breite der Erde bestimmen lassen.

Die planetozentrischen und planetographischen Koordinaten der Sonne können in genau der gleichen Weise berechnet werden. Hierzu sind lediglich die planetozentrischen Erdkoordinaten durch die planetozentrischen Sonnenkoordinaten zu ersetzen.

Die genannten Schritte zur Berechnung der planetographischen Koordinaten fassen wir nun in der Funktion `Rotation` zusammen. Neben der planetographischen Länge und Breite, die üblicherweise in Jahrbüchern angegeben werden, liefert die Funktion auch die planetozentrische Breite zurück, die später noch im Zusammenhang mit der Helligkeitsberechnung benötigt wird. Übergibt man anstelle der planetozentrischen Koordinaten der Erde die planetozentrischen Koordinaten der Sonne, so erhält man ganz analog die planetographische Länge und Breite der Sonne.

```

//-----
// Rotation: Berechnet die planetographischen Koordinaten
//   r      Planetozentrische Koordinaten der Erde bezogen auf Erdaequator
//          und Aequinoktium J2000
//   E      Transformationsmatrix vom geozentrischen, auf den mittleren
//          Aequator und das Aequinoktium J2000 bezogenen Koordinatensystem
//          zum koerperfesten aequatorialen Bezugssystem
//   Sense  Rotationsrichtung (direkt oder retrograd)
//   f      Geometrische Abplattung
//   lon    Planetographische Laenge der Erde in [rad]
//   lat_g  Planetographische Breite der Erde in [rad]
//   lat_c  Planetozentrische Breite der Erde in [rad]
//-----
void Rotation ( const Vec3D& r, const Mat3D& E, RotationType Sense, double f,
                double& lon, double& lat_g, double& lat_c )
{
    Vec3D s;
    // Planetozentrische Breite und Laenge
    s = E*r;  lat_c=s[theta]; lon=s[phi];
    // Planetographische Breite und Laenge
    if (Sense==Direct) lon=-lon; lon=Modulo(lon,2*pi);
    lat_g = atan ( tan(lat_c) / ((1.0-f)*(1.0-f)) );
}

```

7.2 Beleuchtungsverhältnisse

Neben der Beschreibung des aktuellen Rotationszustandes interessiert sich der Beobachter für die Beleuchtungsverhältnisse der Planeten. Diese werden durch die Phase, den Positionswinkel der Sonne und die Helligkeit des Planeten beschrieben.

7.2.1 Phase und Elongation

Da die Planeten unseres Sonnensystems nicht selbst leuchten, sondern von der Sonne bestrahlt werden, ist in der Regel nur ein Teil der von der Erde aus sichtbaren Planetenscheibe beleuchtet. Ein Maß für die Größe des beleuchteten Teils stellt der Winkel zwischen den planetozentrischen Richtungsvektoren zur Sonne und zur Erde dar, der auch als *Phasenwinkel* i bezeichnet wird. Einem Phasenwinkel von $i = 0^\circ$ entspricht eine voll beleuchtete Planetenscheibe, während sich der Himmelskörper bei $i = 90^\circ$ mit Halbphase zeigt.

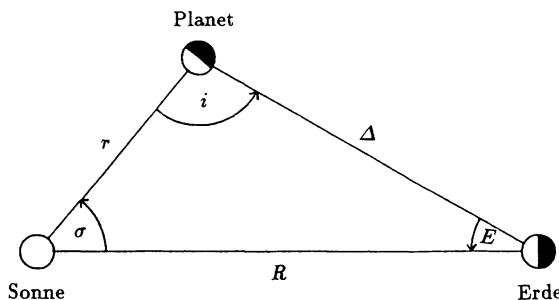


Abb. 7.4. Phasenwinkel i und Elongation E

Bezeichnen wir im Dreieck Sonne-Erde-Planet (Abb. 7.4) die Entfernung des Planeten von der Sonne mit r , die der Erde von der Sonne mit R und den Abstand des Planeten von der Erde mit Δ , so erhalten wir den Phasenwinkel i aus dem ebenen Cosinussatz

$$\cos i = \frac{\Delta^2 + r^2 - R^2}{2r\Delta} . \quad (7.12)$$

Der beleuchtete Anteil der Planetenscheibe, die sogenannte *Phase*, ergibt sich daraus mit der Beziehung

$$k = \frac{1 + \cos i}{2} . \quad (7.13)$$

Zu beachten ist dabei, daß k sich nicht auf die Kugeloberfläche sondern auf die Fläche des scheinbaren Planetenscheibchens bezieht.

Analog zur Berechnung des Phasenwinkels folgt aus dem Cosinussatz

$$\cos E = \frac{\Delta^2 + R^2 - r^2}{2R\Delta} \quad (7.14)$$

im Dreieck Sonne-Erde-Planet auch die *Elongation* E der Sonne, also der vom Standpunkt der Erde aus gemessene Winkelabstand zwischen Sonne und Planet.

Die Elongation gestattet eine einfache Abschätzung, ob der Planet beobachtet werden kann oder ob er für eine Sichtbarkeit zu sehr in der Nähe der Sonne am Himmel steht.

Die Gleichungen zur Berechnung der Elongation und der Phasengestalt der Planeten fassen wir nun in der Funktion `Illum` zusammen. Als Eingabewerte sind die heliozentrischen Koordinaten des Planeten und der Erde zu übergeben. Sie müssen sich auf ein gemeinsames Koordinatensystem und dasselbe Äquinoktium beziehen.

```
-----
// Illum: Berechnet Beleuchtungsparameter der Planeten
// r           Heliozentrischer Ort des Planeten
// r_Earth     Heliozentrischer Ort der Erde
// Elong       Elongation in [rad]
// phi         Phasenwinkel in [rad]
// k           Phase
// Beachte: Koordinatensystem und Epoche fuer r und r_Earth muessen
// uebereinstimmen
-----
void Illum ( const Vec3D& r, const Vec3D& r_Earth,
             double& Elong, double& phi, double& k )
{
    Vec3D   r_geoc;
    double  R,RE,D,c_phi;
    // Geozentrischer Ort des Planeten
    r_geoc = r - r_Earth;
    // Entfernungen
    R = Norm(r);          // Entfernung Sonne-Planet
    RE = Norm(r_Earth);    // Entfernung Sonne-Erde
    D = Norm(r_geoc);     // Entfernung Erde-Planet
    // Elongation, Phasenwinkel und Phase
    Elong = acos ( (D*D + RE*RE - R*R) / (2.0*D*RE) );
    c_phi = (D*D + R*R - RE*RE) / (2.0*D*R);
    phi   = acos (c_phi);
    k     = 0.5*(1.0+c_phi);
}
```

7.2.2 Der Positionswinkel der Sonne

Da sich die Planeten nicht genau in der Ebene der Ekliptik bewegen, fällt das Sonnenlicht nur selten aus exakt westlicher oder östlicher Richtung auf ihre Oberfläche. Die Trennlinie zwischen beleuchteter und dunkler Hemisphäre eines Planeten (der Terminator) verläuft deshalb in der Regel nicht durch die beiden Pole.

Die Orientierung der Lichtphase zur Nordrichtung des Beobachters ist durch den Positionswinkel ϑ_{\odot} der Sonne, der gelegentlich auch als Beleuchtungswinkel bezeichnet wird, eindeutig beschrieben (Abb. 7.5). Mit Hilfe der bereits bekannten Beziehung (7.4) für den Positionswinkel der Rotationsachse kann auch der Positionswinkel der Sonne berechnet werden, wobei anstelle des Richtungsvektors der Achse der Richtungsvektor vom Planeten zur Sonne eingesetzt wird.

Wir können demzufolge die Funktion `PosAng` zur Berechnung des Beleuchtungswinkels verwenden. Dazu übergeben wir die geozentrischen äquatorialen Ko-

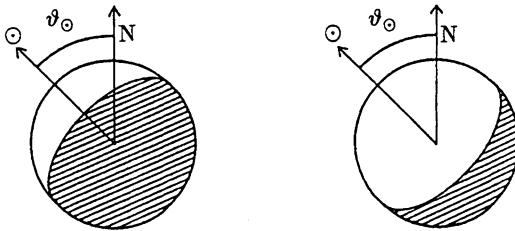


Abb. 7.5. Positionswinkel der Sonne bei verschiedenen Phasenwinkeln

ordinaten des Planeten für den Vektor \mathbf{r} und die planetozentrischen äquatorialen Koordinaten der Sonne als Richtungsvektor \mathbf{d} . Diese ergeben sich durch Vorzeichenumkehr aus den heliozentrischen Planetenkoordinaten, die im Rechengang ohnehin bestimmt werden. Da der Positionswinkel üblicherweise auf die aktuelle Nordrichtung bezogen wird, sind bei der Berechnung die Sonnen- und Planetenkoordinaten ebenfalls auf das Äquinoktium des Datums zu beziehen.

7.2.3 Scheinbare Helligkeit

Unter der scheinbaren Helligkeit eines Planeten verstehen wir die in Größenklassen ausgedrückte Gesamthelligkeit des von der Erde aus sichtbaren beleuchteten Teils. Sie hängt neben der Entfernung des Planeten von der Sonne und der Erde auch von seiner Größe und der Beschaffenheit seiner Oberfläche oder Atmosphäre ab. So beträgt die scheinbare Helligkeit $V(1, 0)$, die ein Planet in einer Entfernung von $r = \Delta = 1 \text{ AE}$ von Sonne und Erde bei einem Phasenwinkel von $i = 0^\circ$ hätte, je nach Durchmesser und Rückstrahlvermögen zwischen -0^m4 (Merkur) und -9^m4 Größenklassen (Jupiter).

Mit zunehmender Entfernung von der Sonne verringert sich die Beleuchtungsstärke am Ort des Planeten umgekehrt proportional zum Quadrat des Abstandes. Ebenso nimmt der beobachtete Strahlungsfluß bei gegebener Entfernung des Planeten von der Sonne quadratisch mit seiner Entfernung von der Erde ab. Berücksichtigt man ferner, daß eine Abnahme des Strahlungsflusses um den Faktor 100 einer Änderung der Helligkeit um +5 Größenklassen entspricht, dann ergibt sich für die scheinbare Helligkeit eines Planeten der Zusammenhang

$$m = V(1, 0) + 5 \log \left(\frac{r \cdot \Delta}{\text{AE}^2} \right) + \Delta m(i) \quad . \quad (7.15)$$

Der Term $\Delta m(i)$ modelliert hierbei die Phasenabhängigkeit der Helligkeit. Diese entsteht durch die geringere beleuchtete Fläche bei abnehmender Phase und durch die Winkelabhängigkeit der Streuung und Reflexion des Lichtes. $\Delta m(i)$ wird meist durch ein aus Beobachtungsdaten ermitteltes Potenzgesetz beschrieben.

Bei Saturn ist zusätzlich der Beitrag des Ringsystems zur Planetenhelligkeit zu berücksichtigen, die als Funktion der planetozentrischen Breite φ'_S der Sonne und der planetozentrischen Längendifferenz $\lambda'_S - \lambda'_E$ von Sonne und Erde modelliert werden kann.

Ausdrücke für die Bezugshelligkeiten $V(1,0)$ und die phasenabhängigen Helligkeitsanteile $\Delta m(i)$ der Planeten sind in der Tabelle 7.3 zusammengestellt. Die Funktion `Bright` verwendet diese Zahlenwerte zur Bestimmung der scheinbaren Helligkeiten der Planeten.

Tabelle 7.3. Scheinbare Helligkeiten der Planeten

Planet	$V(1,0)$	$\Delta m(i)$
Merkur	$-0^m42 + 3^m80 \cdot \left(\frac{i}{100^\circ}\right) - 2^m73 \cdot \left(\frac{i}{100^\circ}\right)^2 + 2^m00 \cdot \left(\frac{i}{100^\circ}\right)^3$	
Venus	$-4^m40 + 0^m09 \cdot \left(\frac{i}{100^\circ}\right) + 2^m39 \cdot \left(\frac{i}{100^\circ}\right)^2 - 0^m65 \cdot \left(\frac{i}{100^\circ}\right)^3$	
Erde	-3^m86	
Mars	$-1^m52 + 1^m60 \cdot \left(\frac{i}{100^\circ}\right)$	
Jupiter	$-9^m40 + 0^m50 \cdot \left(\frac{i}{100^\circ}\right)$	
Saturn	$-8^m88 - 2^m60 \cdot \sin \varphi'_\odot + 1^m25 \cdot \sin \varphi'_\odot ^2 + 4^m40 \cdot \lambda'_\oplus - \lambda'_\odot $	
Uranus	-7^m19	
Neptun	-6^m87	
Pluto	-1^m0	

```
-----
// Bright: Berechnet die visuelle Helligkeit eines Planeten
// Planet   Identifiziert den Planeten (s. APC_Planets.h)
// r        Heliozentrische Entfernung des Planeten in [AE]
// Delta   Entfernung des Planeten von der Erde in [AE]
// phi     Phasenwinkel in [rad]
// lat      Planetozentrische Breite der Sonne in [rad]
// Dlong   Differenz der planetozentrischen Laengen von Sonne und Erde
//          in [rad]
// <return> Helligkeit des Planeten in Groessenklassen [mag]
// Beachte: lat und Dlong muessen nur fuer Saturn angegeben werden und koennen
//           ansonsten weggelassen werden
-----
double Bright ( PlanetType Planet, double r, double Delta, double phi,
                double lat=0.0, double Dlong=0.0 );
double Bright ( PlanetType Planet, double r, double Delta, double phi,
                double lat, double Dlong )
{
    double p, sd, dl, mag;
    p = Deg*phi/100.0;
    switch ( Planet ) {
        case Sun:      return 0.0;
        case Mercury:  mag = -0.42+(3.80-(2.73-2.00*p)*p)*p; break;
        case Venus:    mag = -4.40+(0.09+(2.39-0.65*p)*p)*p; break;
        case Earth:    mag = -3.86;           break;
        case Mars:     mag = -1.52+1.6*p; break;
        case Jupiter:  mag = -9.40+0.5*p; break;
        case Saturn:
            sd = fabs(sin(lat));
            dl = fabs(Modulo(Deg*Dlong+180.0,360.0)-180.0)/100.0;
            mag = -8.88+(3.60-(2.53-2.00*p)*p)*p;
            if ( sd > 0.0 ) mag -= 0.5*(sd-0.5);
            if ( dl > 0.0 ) mag -= 0.5*(dl-0.5);
            break;
        default:       mag = -9.40+0.5*p; break;
    }
}
```

```

mag = -8.88 - 2.60*sd + 1.25*sd*sd + 4.40*dl;
break;
case Uranus: mag = -7.19; break;
case Neptune: mag = -6.87; break;
case Pluto: mag = -1.0;
}
return ( mag + 5.0*log10(r*Delta) );
}

```

7.2.4 Scheinbarer Durchmesser

Der scheinbare Durchmesser eines Planeten ist der Sichtwinkel, unter dem der Planet von der Erde aus erscheint. Er ergibt sich mit dem Äquatorradius $R_{\text{Äq}}$ des Planeten bei einem Abstand Δ von der Erde zu

$$\varnothing_{\text{Äq}} = 2 \cdot \arcsin \left(\frac{R_{\text{Äq}}}{\Delta} \right) . \quad (7.16)$$

Bei abgeplatteten Planeten ist zwischen dem scheinbaren Durchmesser am Äquator und am Pol zu unterscheiden. Der scheinbare Poldurchmesser ist dabei stets kleiner als der scheinbare Äquatordurchmesser und hängt neben der Abplattung auch von der planetozentrischen Breite φ'_{\oplus} der Erde ab. Für kleine Werte der Abplattung f kann der scheinbare Poldurchmesser in erster Näherung aus der Beziehung

$$\varnothing_{\text{Pol}} = (1 - f \cos^2 \varphi'_{\oplus}) \varnothing_{\text{Äq}} \quad (7.17)$$

bestimmt werden.

7.3 Das Programm PHYS

Das Programm **Phys** berechnet eine Übersicht der physischen Ephemeriden der großen Planeten und der Sonne. Dazu verwendet es die bereits vorgestellten Funktionen für die Bestimmung der planetographischen Koordinaten der Erde, der Beleuchtungsverhältnisse und des Positionswinkels des Nordpols der Rotationsachse.

Die notwendigen Koordinaten der Himmelskörper werden von der Funktion **PertPosition** unter Verwendung analytischer Reihenentwicklungen für das Äquinoktium des Datums berechnet. Die Auswirkungen der Nutation und Aberration sind klein genug, um sie im Rahmen der angestrebten Genauigkeit für praktische Beobachtungen vernachlässigen zu können. Wesentlich ist dagegen die Berücksichtigung der Lichtlaufzeit vom Planeten zur Erde. Dazu wird in einem ersten Schritt die geometrische Erdentfernung des Himmelskörpers bestimmt und daraus die Lichtlaufzeit ermittelt. Im weiteren Verlauf der Rechnung werden dann die für den Zeitpunkt der Lichtaussendung bestimmten Koordinaten und Rotationswinkel zugrundegelegt.

Die Präzession auf das aktuelle Äquinoktium des Datums braucht bei der Bestimmung der planetographischen bzw. heliographischen Koordinaten der Erde und für das Ermitteln der Beleuchtungsverhältnisse nicht berücksichtigt werden. Diese Größen ergeben sich aus den relativen Positionen der Himmelskörper und

sind damit von der Wahl des Koordinatensystems unabhängig. Lediglich für die Berechnung der Positionswinkel, die sich auf die aktuelle Nordrichtung und damit auf das Äquinoktium des Datums beziehen, ist eine entsprechende Koordinatentransformation erforderlich.

Nach der Berechnung der physischen Ephemeriden der Planeten werden für die Sonne der scheinbare Durchmesser, der Positionswinkel der Achse und die heliographischen Koordinaten der Erde bestimmt. Die Orientierung des planetenfesten Koordinatensystems relativ zum Erdäquator und Frühlingspunkt wird in `Orient` bestimmt und direkt an `Rotation` übergeben. Die geozentrischen Sonnenkoordinaten (wie sie auch zur Berechnung des Positionswinkels der Achse benötigt werden) ergeben sich durch Vorzeichenumkehr aus den heliozentrischen Koordinaten der Erde.

Weiterhin ist noch zu beachten, daß die heliographische Länge entgegengesetzt der planetographischen Längendefinition gezählt wird. Die heliographische Länge der Erde nimmt also laufend ab. Darauf hinaus gibt man einer allgemeinen Konvention zufolge den Positionswinkel der Sonnenachse im Intervall $-180^\circ \dots 180^\circ$ an, was durch entsprechende Anweisungen im Programm berücksichtigt wird.

```
//-----
// Datei: Phys.cpp
// Zweck: Physische Ephemeriden der grossen Planeten und der Sonne
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//-----

#include <cmath>
#include <iomanip>
#include <iostream>

#include "APC_Const.h"
#include "APC_Math.h"
#include "APC_Phys.h"
#include "APC_Planets.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"

using namespace std;

//-----
// 
// Hauptprogramm
// 
//-----
void main()
{
    // Namen der Himmelskörper
    const char* Name[] =
    { "Sonne", "Merkur", "Venus", "Erde", "Mars",
      "Jupiter", "Saturn", "Uranus", "Neptun", "Pluto" };
}
```

```

// Variablen
PlanetType   Planet;
int          iPlanet, year, month, day;
double        hour, T, T0;
Vec3D         r, r_Earth, r_geoc;
double        Delta;
double        R_equ, D_equ, f;
Mat3D         E_I, E_II, E_III;
Mat3D         P;
Vec3D         d;
RotationType Sense;
double        long_I, long_II, long_III, lat_g, lat_c;
double        long_Sun, latg_Sun, latc_Sun;
double        Elong, phi, k, mag;
double        PosAng_Ax, PosAng_Sun;

// Titel
cout << endl
    << " PHYS: Physische Ephemeriden der Planeten und der Sonne" << endl
    << "           (c) 1999 Oliver Montenbruck, Thomas Pfleger      " << endl
    << endl;
// Abfrage der Eingabedaten
cout << " Berechnungsdatum (JJJJ MM TT HH.HHH) ... ";
cin >> year >> month >> day >> hour; cin.ignore(81,'\'n');
T = (Mjd(year,month,day)+hour/24.0-MJD_J2000)/36525.0;
// Vorspann
cout << endl
    << "          D      V      i      PW(S)      PW(A)"
    << "    L(I)  L(II)  L(III)      B" << endl
    << "          , \     mag      o      o      o "
    << "          o      o      o      o" << endl;

// Praezession (Aequator und Aequinoktium des Datums nach J2000)
P = PrecMatrix_Equ(T,T_J2000);
// Heliozentrische Position der Erde, mittlerer Aequator und Aequinoktium
// des Datums
r_Earth = Ecl2EquMatrix(T) * PertPosition(Earth,T);

// Schleife ueber die Himmelskoerper
for (iPlanet=Sun; iPlanet<=Pluto; iPlanet++) {
    Planet = (PlanetType) iPlanet;
    if (Planet==Earth) continue; // Erde ueberspringen
    // Geometrische geozentrische Position und Lichtlaufzeit
    r = Ecl2EquMatrix(T) * PertPosition(Planet,T);
    Delta = Norm(r-r_Earth);
    T0 = T - (Delta/c_light)/36525.0,
    // Planetenposition zum Zeitpunkt der Lichtaussendung
    r = Ecl2EquMatrix(T) * PertPosition(Planet,T0);
    // Geozentrische Position
    r_geoc = r - r_Earth;
    // Scheinbare Halbmesser (in ["])
    Shape ( Planet, R_equ,f );
    D_equ = Arcs*2.0*asin(R_equ/(Delta*AU));
    // Transformation vom mittleren Aequator und Aequinoktium des
    // Datums zum koerperfesten Bezugssystem

```

```

Orient ( Planet, Sys_I ,T0, E_I , Sense ); E_I = E_I * P;
Orient ( Planet, Sys_II ,T0, E_II , Sense ); E_II = E_II * P;
Orient ( Planet, Sys_III,T0, E_III, Sense ); E_III = E_III * P;
// Planetozentrische Laenge und Breite der Erde
Rotation ( -r_geoc, E_I, Sense, f, long_I, lat_g,lat_c );
Rotation ( -r_geoc, E_II, Sense, f, long_II, lat_g,lat_c );
Rotation ( -r_geoc, E_III, Sense, f, long_III,lat_g,lat_c );
// Positionsinkel der Rotationsachse
PosAng_Ax = PosAng ( r_geoc, Row(E_I,z) );
// Spezifische Berechnungen fuer Sonne und Planeten
if (Planet==Sun) {
    // Sonnenspezifische Definition fuer Laenge und Positionswinkel
    // der Rotationsachse
    long_I = 2.0*pi-long_I;
    if (PosAng_Ax>pi) PosAng_Ax-=2.0*pi;
}
else {
    // Beleuchtung und scheinbare Helligkeit
    Illum ( r, r_Earth, Elong,phi,k );
    if (Planet==Saturn) {
        // Planetozentrische Laenge und Breite der Sonne
        Rotation ( -r, E_I, Sense, f, long_Sun, latg_Sun, latc_Sun );
        mag = Bright ( Planet, Norm(r), Norm(r_geoc), phi,
                        latc_Sun, long_Sun-long_I );
    }
    else
        mag = Bright ( Planet, Norm(r),Norm(r_geoc), phi );
    // Positionswinkel der Sonne
    PosAng_Sun = PosAng ( r_geoc, -r );
};

// Ausgabe
if (Planet==Sun)
    cout << " " << left << setw(6) << Name[Planet] << right << fixed
        << setprecision(2) << setw(8) << Angle(D_equ/60.0,DMMm)
        << setprecision(2) << setw(29) << Deg*PosAng_Ax;
else
    cout << " " << left << setw(9) << Name[Planet] << right << fixed
        << setprecision(2) << setw(5) << D_equ
        << setprecision(1) << setw(6) << mag
        << setprecision(1) << setw(7) << Deg*phi
        << setprecision(2) << setw(8) << Deg*PosAng_Sun
        << setprecision(2) << setw(8) << Deg*PosAng_Ax;
// Planetographische Laenge(n) der Erde ausgeben
switch (Planet) {
    // Himmelskoerper, fuer die nur Rotationssystem I definiert ist
    case Sun: case Mercury: case Venus: case Mars: case Pluto:
        cout << setprecision(2) << setw(8) << Deg*long_I
            << setw(14) << " ";
        break;
    case Jupiter:
        cout << setprecision(2) << setw(8) << Deg*long_I
            << setprecision(2) << setw(7) << Deg*long_II
            << setprecision(2) << setw(7) << Deg*long_III;
        break;
}

```

```

case Saturn:
    cout << setprecision(2) << setw(8)  << Deg*long_I
        << setprecision(2) << setw(14) << Deg*long_III;
    break;
// Himmelskoerper, fuer die nur Rotationssystem III definiert ist
case Uranus: case Neptune:
    cout << setprecision(2) << setw(22) << Deg*long_III;
}
cout << setprecision(2) << setw(8)  << Deg*lat_g << endl;
} // Ende der Schleife ueber die Himmelskoerper
}

```

Als Beispiel sollen die physischen Ephemeriden für den 30. März 1993 um 0^h Ephemeridenzeit ermittelt werden. Das einzugebende Berechnungsdatum ist wie immer kursiv hervorgehoben.

Angezeigt werden der scheinbare Äquator Durchmesser D in Bogensekunden, die visuelle Helligkeit V in Größenklassen, der Phasenwinkel i, sowie die Positionswinkel der Sonne PW(S) und des Nordpols PW(A). Ferner werden die planetographischen Koordinaten der Erde in den für den jeweiligen Planeten gebräuchlichen Rotationssystemen angegeben.

PHYS: Physische Ephemeriden der Planeten und der Sonne
(c) 1999 Oliver Montenbruck, Thomas Pfleger

Berechnungsdatum (JJJJ MM TT HH.HHH) ... 1999 08 30 00.0

	D	V	i	PW(S)	PW(A)	L(I)	L(II)	L(III)	B
	,	"	mag	o	o	o	o	o	o
Sonne	31	40.64			20.43	203.33			7.15
Merkur	5.30	-1.4	31.6	119.79	21.10	328.45			5.31
Venus	55.62	-4.2	156.5	79.44	17.13	334.31			8.68
Mars	7.95	0.3	44.2	283.48	37.85	29.70			12.81
Jupiter	45.22	-2.7	10.0	71.42	339.34	329.80	176.12	24.39	3.78
Saturn	18.73	0.2	5.9	74.63	359.41	93.75		118.66	-25.77
Uranus	3.69	5.7	1.1	251.46	262.90			155.83	-34.84
Neptun	2.29	7.9	1.1	257.45	355.90			256.25	-28.71
Pluto	0.14	13.8	1.9	279.11	75.60	311.55			-22.30

Venus steht etwa in unterer Konjunktion einige Grad südlich der Sonne und zeigt eine sehr große und schmale Sichel ($i = 156^\circ 5$), wie sie unter sehr günstigen Beobachtungsbedingungen bereits am Taghimmel gesehen werden kann. Das Sonnenlicht kommt ziemlich genau aus nördlicher Richtung ($\vartheta_\odot = 79^\circ 4$).

Bei Jupiter fallen der große scheinbare Durchmesser und seine Helligkeit auf. Jupiter steht zu diesem Zeitpunkt am Umkehrpunkt seiner Schleifenbewegung und steuert auf die rund zwei Monate später stattfindende Opposition zu.

Mars dagegen hat seine Opposition schon hinter sich und zeigt eine deutliche Phase ($i = 44^\circ 2$). Gleichzeitig macht sein kleiner Durchmesser von knapp acht Bogensekunden die Erkennung von Strukturen auf der Oberfläche schwierig.

8. Die Mondbahn

8.1 Allgemeine Beschreibung der Mondbahn

Die Bewegung des Mondes wird vornehmlich durch zwei Körper bestimmt, nämlich durch Erde und Sonne. Betrachtet man die Anziehungskräfte, die auf den Mond einwirken, so stellt man fest, daß nicht die Erde als nächster Nachbar, sondern die weiter entfernte Sonne die Mondbahn am stärksten beeinflußt. Obwohl die Stärke der Gravitationskraft quadratisch mit der Entfernung abnimmt, übertrifft die Anziehung der Sonne aufgrund der hohen Masse die der Erde. Mit den entsprechenden Zahlenwerten (Entfernung Erde-Mond $r \approx 380000$ km, Entfernung Sonne-Mond $R \approx 150$ Mio. km, Verhältnis Sonnenmasse zu Erdmasse $M/m \approx 330000$) sieht man, daß die Anziehungskraft der Sonne etwa doppelt so groß ist, wie die der Erde:

$$\frac{F_{\odot}}{F_{\oplus}} = \frac{M}{R^2} / \frac{m}{r^2} \approx 2 \quad .$$

Die Summe beider Kräfte hat damit unabhängig von der gegenseitigen Orientierung von Sonne, Erde und Mond immer eine resultierende Komponente, die in Richtung der Sonne und niemals von ihr weg zeigt. Damit läßt sich die räumliche Bewegung des Mondes als eine rund 150 Mio. km große Ellipse um die Sonne beschreiben, der kleinen monatlichen Schwankungen überlagert sind. Da die Gravitationskraft nie von der Sonne weg weist, ist die Bahn trotz dieser Schwankungen immer zur Sonne hin gekrümmmt.

Obwohl die Mondbahn also hauptsächlich durch die Anziehungskraft der Sonne bestimmt wird, erscheint es nicht unbedingt ratsam, sie in heliozentrischen Koordinaten zu beschreiben. Schließlich sind wir ja letztendlich an der Bewegung des Mondes um die Erde interessiert. Da Erde und Mond relativ eng benachbart sind, wirken auf sie fast gleiche Gravitationskräfte von seiten der Sonne. Wären diese Kräfte exakt gleich groß, dann hätte die Sonne überhaupt keinen Einfluß auf die *gegenseitige* Lage von Erde und Mond. Sie wäre dann alleine für deren gemeinsame jährliche Bahn um die Sonne verantwortlich, während die Anziehungskraft der Erde die monatliche Bewegung des Mondes bestimmen würde. Die Differenz der Sonnenanziehung auf Erde und Mond ist nur etwa r/R -mal so groß wie die Anziehung selbst und damit rund 200mal kleiner als die Kraft, die von seiten der Erde auf den Mond wirkt. Die zweckmäßigste Beschreibung der Mondbahn erhält man deshalb in einem geozentrischen Bezugssystem. Der Mond bewegt sich darin auf einer monatlichen Keplerbahn um die Erde, die von der Sonne mehr oder weniger stark gestört wird.

Die mittlere Bahnellipse hat eine Exzentrizität von $e = 0.055$ und ist rund $5^\circ 1$ gegen die Ekliptik geneigt. Die Umlaufszeit von Perigäum zu Perigäum (sogenannter *anomalistischer* Monat) beträgt im Durchschnitt 27.55 Tage. Dies sind knapp zwei Tage weniger als die Zeit, die zwischen zwei aufeinanderfolgenden Neu- oder Vollmondphasen vergeht (*synodischer* Monat). Dieser Unterschied erklärt sich aus der scheinbaren Bewegung der Sonne vor dem Himmelshintergrund, die im Monat rund 30° beträgt. Nach einem Umlauf um die Erde benötigt der Mond etwa zwei Tage, um diesen Vorsprung der Sonne einzuholen und wieder die gleiche Phase zu zeigen.

Aufgrund der exzentrischen Bahn weicht der Mond um bis zu $6^\circ 3$ von seinem mittleren Ort ab. Diese auffallendste Unregelmäßigkeit wird als *große Ungleichheit* bezeichnet. Sie hat noch nichts mit den Störungen durch die Sonne zu tun, sondern ist allein auf die elliptische Bahnform zurückzuführen. Der Betrag der großen Ungleichheit folgt aus der Mittelpunktsgleichung (6.1), die bereits bei der Reihenentwicklung der Planetenbahnen erwähnt wurde. Schon Ptolemäus wies aber im Almagest darauf hin, daß sich die Mondbahn damit alleine noch nicht befriedigend beschreiben läßt. Er bemerkte als erster die *Evection*, eine Störung von $1:3$, die von der gegenseitigen Stellung von Sonne und Mond abhängt. Wesentlich später wurden von Tycho de Brahe aufgrund seiner genauen Beobachtungen zwei weitere Störungen (*Variation* und *jährliche Gleichung*) entdeckt, bevor Newton mit der Aufstellung des Gravitationsgesetzes die Grundlage für ein theoretisches Verständnis der Mondbewegung schuf. Im Jahre 1770 wurden von Mayer erstmals Mondtafeln veröffentlicht, die genau genug waren, um eine Orts- und Zeitbestimmung auf See zu gestatten. Heute sind insgesamt über tausend einzelne periodische Störungsterme bekannt.

Neben diesen mehr oder minder großen periodischen Schwankungen der Mondbahn bewirkt der Einfluß der Sonne auch eine Reihe säkularer Effekte, deren wichtigster eine langsame Drehung der Mondbahnebene ist. In 18.6 Jahren dreht sich bei nahezu fester Bahnneigung die Knotenlinie, also die Schnittlinie von Mondbahn und Ekliptik, um volle 360° . Diese Knotenwanderung erfolgt *rückläufig*, das heißt, die Länge des aufsteigenden Knotens *verringert* sich gegenüber dem Frühlingspunkt jährlich um etwa 20° . Der Grund für diese Verlagerung liegt darin, daß die Anziehungskraft der Sonne ständig versucht, den Mond in die Ekliptik zu ziehen und die Symmetrieachse der Bahn aufzurichten. Diese weicht jedoch wie eine Kreiselachse der einwirkenden Kraft aus und beginnt eine langsame Präzessionsbewegung, die sie in den besagten 18.6 Jahren einmal um den Pol der Ekliptik herumführt.

Die Lage des Perigäums, also die räumliche Orientierung der großen Halbachse der Bahn, ist ebenfalls veränderlich. Jedes Jahr verschiebt sich das Perigäum in der Bahnebene (rechtläufig) um über 40° . Ein voller Umlauf dauert etwa 8.85 Jahre.

Die verschiedenen Größen für die mittlere Bewegung von Sonne und Mond, die im Rahmen der Mondtheorie Verwendung finden, sind mit den dort gebräuchlichen Bezeichnungen ($T = (JD - 2451545)/36525$):

- die mittlere Länge des Mondes (L_0)

$$\begin{aligned}L_0 &= 218^\circ 31617 + 481267^\circ 88088 \cdot T - 4\overset{''}{.}06 \cdot T^2 \\&= 0^\circ 60643382 + 1336^\circ 85522467 \cdot T - 0^\circ 00000313 \cdot T^2 ,\end{aligned}\quad (8.1)$$

- die mittlere Anomalie des Mondes (l)

$$\begin{aligned}l &= 134^\circ 96292 + 477198^\circ 86753 \cdot T + 33\overset{''}{.}25 \cdot T^2 \\&= 0^\circ 37489701 + 1325^\circ 55240982 \cdot T + 0^\circ 00002565 \cdot T^2 ,\end{aligned}\quad (8.2)$$

- die mittlere Anomalie der Sonne (l')

$$\begin{aligned}l' &= 357^\circ 52543 + 35999^\circ 04944 \cdot T - 0\overset{''}{.}58 \cdot T^2 \\&= 0^\circ 99312619 + 99^\circ 99735956 \cdot T - 0^\circ 00000044 \cdot T^2 ,\end{aligned}\quad (8.3)$$

- der mittlere Abstand des Mondes vom aufsteigenden Knoten (F)

$$\begin{aligned}F &= 93^\circ 27283 + 483202^\circ 01873 \cdot T - 11\overset{''}{.}56 \cdot T^2 \\&= 0^\circ 25909118 + 1342^\circ 22782980 \cdot T - 0^\circ 00000892 \cdot T^2 ,\end{aligned}\quad (8.4)$$

- und die mittlere Elongation des Mondes (D), also die Differenz der mittleren Längen von Sonne und Mond,

$$\begin{aligned}D &= 297^\circ 85027 + 445267^\circ 11135 \cdot T - 5\overset{''}{.}15 \cdot T^2 \\&= 0^\circ 82736186 + 1236^\circ 85308708 \cdot T - 0^\circ 00000397 \cdot T^2 .\end{aligned}\quad (8.5)$$

Die einzelnen Werte sind im Gradmaß und in Einheiten eines Umlaufs ($1^r = 360^\circ$) angegeben. L_0 bezieht sich auf das mittlere Äquinoktium des Datums, die übrigen Größen sind Winkel, die nicht von der Präzession beeinflußt werden. Die Länge des aufsteigenden Knotens (Ω) wird nicht explizit verwendet. Sie ergibt sich aus der Differenz $\Omega = L_0 - F$.

Die wahre ekliptikale Länge des Mondes (λ) unterscheidet sich von der mittleren Länge durch eine Reihe periodischer Terme, die von den Argumenten l , l' , F und D abhängen:

$$\lambda = L_0 + \Delta\lambda$$

mit

$$\begin{aligned}
 \Delta\lambda = & +22640'' \cdot \sin(l) && (\text{große Ungleichheit}) \\
 & -4586'' \cdot \sin(l - 2D) && (\text{Evektion}) \\
 & +2370'' \cdot \sin(2D) && (\text{Variation}) \\
 & +769'' \cdot \sin(2l) && (\text{große Ungleichheit}) \\
 & -668'' \cdot \sin(l') && (\text{jährliche Gleichung}) \\
 & -412'' \cdot \sin(2F) && (\text{Reduktion auf die Ekliptik}) \\
 & -212'' \cdot \sin(2l - 2D) \\
 & -206'' \cdot \sin(l + l' - 2D) \\
 & +192'' \cdot \sin(l + 2D) \\
 & -165'' \cdot \sin(l' - 2D) \\
 & +148'' \cdot \sin(l - l') \\
 & -125'' \cdot \sin(D) && (\text{parallaktische Gleichung}) \\
 & -110'' \cdot \sin(l + l') \\
 & -55'' \cdot \sin(2F - 2D) .
 \end{aligned}$$

In einer ungestörten Bahn hängt die ekliptikale Breite β über

$$\begin{aligned}
 \sin \beta &= \sin i \cdot \sin u \\
 \beta &\approx i \cdot \sin u
 \end{aligned}$$

von der Bahnneigung i und dem Ort in der Bahn ab. u bezeichnet darin das Argument der Breite, also den Winkel zwischen dem Ortsvektor und der Knotenlinie. Angesichts der Störungen durch die Sonne wird die Breite des Mondes in erster Näherung durch eine leicht abgewandelte Gleichung, nämlich durch

$$\beta \approx 18520'' \sin(S) + N$$

mit $S = F + \Delta S$ berechnet. ΔS setzt sich aus einer Reihe von Termen zusammen, die im wesentlichen mit der Entwicklung von $\Delta\lambda$ übereinstimmen. Beschränkt man sich auf die wichtigsten Unterschiede, dann ist

$$\Delta S \approx \Delta\lambda + 412'' \sin(2F) + 541'' \sin(l') .$$

Die Größe N beinhaltet eine kleine Zahl zusätzlicher Breitenänderungen, die durch eine Schwankung der Bahnneigung verursacht werden:

$$N = -526'' \sin(F - 2D) + 44'' \sin(l + F - 2D) - 31'' \sin(-l + F - 2D) + \dots \quad (8.6)$$

Die bisherige, stark vereinfachte Darstellung soll zunächst als grober Überblick über die Formulierung der Mondtheorie und die Berechnung der ekliptikalnen Länge und Breite des Mondes dienen. Im Programm **MiniMoon**, das in Kap.3 ohne nähere Erläuterungen vorgestellt wurde, sind gerade die bisher besprochenen Gleichungen verarbeitet. Ein Blick auf dieses Programm sollte die einzelnen Rechenschritte noch einmal verdeutlichen.

Anzumerken ist noch, daß für die Berechnung der ekliptikalnen Breite auf die Verwendung einer Reihenentwicklung von β selbst bewußt verzichtet wurde. Eine solche Reihe, deren wichtigste Glieder

$$\begin{aligned}
 \beta = & 18461'' \sin(F) + 1010'' \sin(l + F) + 1000'' \sin(l - F) - 624'' \sin(F - 2D) \\
 & -199'' \sin(l - F - 2D) - 167'' \sin(l + F - 2D) + \dots
 \end{aligned}$$

sind, ist zwar leichter zu verwenden, erfordert aber einen unnötig hohen Rechenaufwand. Dies liegt daran, daß die einzelnen Terme hier nur ungerade Vielfache von F enthalten, während in den Termen der Entwicklung von $\Delta\lambda$ und ΔS immer nur gerade Vielfache von F auftreten. Zu jedem Term in $\Delta\lambda$ gibt es einen gleichartigen Term in ΔS , für den dieselbe Winkelfunktion benötigt wird. Bei entsprechender Organisation müssen für ΔS deshalb keine zusätzlichen trigonometrischen Terme mehr ausgewertet werden. Dagegen haben die Reihen von $\Delta\lambda$ und $\Delta\beta$ keine Terme, die gemeinsam ausgewertet werden könnten.

8.2 Die Brownsche Mondtheorie

Die zu Beginn des Jahrhunderts entwickelte Mondtheorie von E.W. Brown zählt zu den bekanntesten analytischen Formulierungen der Mondbewegung. Sie wird hier in der Fassung der *Improved Lunar Ephemeris* (ILE) des Nautical Almanac Office von 1954 vorgestellt. Von den weit über tausend Störungstermen der Theorie wird allerdings nur ein Bruchteil in der Funktion MoonPos verwendet. Die damit erreichte Genauigkeit liegt bei rund einer Bogensekunde.

Die Rechnung beginnt mit der Bestimmung der mittleren Argumente L, l, l', F und D nach (8.1) bis (8.5). Die Zeit T ist dabei in julianischen Jahrhunderten Ephemeridenzeit seit der Epoche J2000 ausgedrückt. Die mittleren Argumente unterliegen zusätzlich noch kleinen langperiodischen Schwankungen, die als Korrekturen zu den obigen Werten hinzuaddiert werden:

$$\begin{array}{lllll}
 \Delta L_0 & \Delta l & \Delta l' & \Delta F & \Delta D \\
 +0.^{\circ}84 & +2.^{\circ}94 & -6.^{\circ}40 & +0.^{\circ}21 & +7.^{\circ}24 \cdot \sin(2\pi(0.19833 + 0.05611T)) \\
 +0.^{\circ}31 & +0.^{\circ}31 & 0.^{\circ}0 & +0.^{\circ}31 & +0.^{\circ}31 \cdot \sin(2\pi(0.27869 + 0.04508T)) \\
 +14.^{\circ}27 & +14.^{\circ}27 & 0.^{\circ}0 & +14.^{\circ}27 & +14.^{\circ}27 \cdot \sin(2\pi(0.16827 - 0.36903T)) \\
 +7.^{\circ}26 & +9.^{\circ}34 & 0.^{\circ}0 & -88.^{\circ}70 & +7.^{\circ}26 \cdot \sin(2\pi(0.34734 - 5.37261T)) \\
 +0.^{\circ}28 & +1.^{\circ}12 & 0.^{\circ}0 & -15.^{\circ}30 & +0.^{\circ}28 \cdot \sin(2\pi(0.10498 - 5.37899T)) \\
 +0.^{\circ}24 & +0.^{\circ}83 & -1.^{\circ}89 & +0.^{\circ}24 & +2.^{\circ}13 \cdot \sin(2\pi(0.42681 - 0.41855T)) \\
 0.^{\circ}00 & 0.^{\circ}00 & 0.^{\circ}00 & -1.^{\circ}86 & 0.^{\circ}00 \cdot \sin(2\pi(0.14943 - 5.37511T))
 \end{array} \quad (8.7)$$

Mit den so verbesserten mittleren Argumenten der Mondbahn werden insgesamt fünf Reihen von Störungstermen berechnet:

- | | |
|---------------------------|--|
| $\Delta\lambda$ | Störungen der ekliptikalnen Länge, |
| $\Delta S, N, \gamma_1 C$ | Störungen der ekliptikalnen Breite und |
| $\Delta \sin \Pi$ | Störungen der Parallaxe . |

Alle diese Reihen haben die gemeinsame Struktur

$$\left\{ \begin{array}{l} \Delta\lambda, \Delta S, N \\ \gamma_1 C, \Delta \sin \Pi \end{array} \right\} = \sum_n \left\{ \begin{array}{l} a_n, b_n, c_n \\ d_n, e_n \end{array} \right\} \cdot \left\{ \begin{array}{l} \sin \\ \cos \end{array} \right\} (p_n l + q_n l' + r_n F + s_n D) .$$

Das Tupel (p, q, r, s) beschreibt dabei die Abhängigkeit eines einzelnen Summanden von l, l', F und D und damit die Periodizität des Terms. Für die Rechnung faßt man zweckmäßigerweise alle Terme gleicher Charakteristik zusammen. Aus

der gesamten Tabelle der Störungsterme ist hier nur ein kleiner Ausschnitt wiedergegeben:

$\Delta\lambda$	ΔS	$\gamma_1 C$	$\Delta \sin \Pi$	p	q	r	s
+22639."500	+22609."07	+0."079	+186."5398	+1	0	0	0
-4586."465	-4578."13	-0."077	+34."3117	+1	0	0	-2
+2369."912	+2373."36	+0."601	+28."2333	0	0	0	+2
+769."016	+767."96	+0."107	+10."1657	+2	0	0	0
-668."146	-126."98	-1."302	-0."3997	0	+1	0	0
-411."608	-0."20	+0."000	-0."0124	0	0	+2	0

Die Reihe für die Größe N kann aufgrund der unterschiedlichen Charakteristik nicht mit den übrigen Reihen zusammen berechnet werden. Sie enthält aber auch nur einige wenige Glieder.

Angesichts der großen Zahl von Störungstermen lohnt es sich durchaus, einige Mühe auf die Berechnung der auftretenden Winkelfunktionen zu verwenden. Die Berechnung einer Sinus- oder Cosinusfunktion ist – verglichen mit elementaren Operationen wie den vier Grundrechenarten – relativ aufwendig und rechenezeitintensiv. Es ist aber gar nicht notwendig, jede Winkelfunktion explizit auszuwerten, wenn man auf die Additionstheoreme

$$\begin{aligned}\cos(\alpha_1 + \alpha_2) &= \cos \alpha_1 \cos \alpha_2 - \sin \alpha_1 \sin \alpha_2 \\ \sin(\alpha_1 + \alpha_2) &= \sin \alpha_1 \cos \alpha_2 + \cos \alpha_1 \sin \alpha_2\end{aligned}\quad (8.8)$$

zurückgreift. Mit Hilfe der kurzen Funktion AddThe

```
void AddThe(double c1, double s1, double c2, double s2, double& c, double& s)
{ c=c1*c2-s1*s2; s=s1*c2+c1*s2; }
```

kann man zunächst Sinus und Cosinus von Vielfachen der mittleren Argumente berechnen:

$$\text{Cos}[j+o][i] = \begin{cases} \cos(jl) & (i = 0) \\ \cos(jl') & (i = 1) \\ \cos(jF) & (i = 2) \\ \cos(jD) & (i = 3) \end{cases} \quad \text{Sin}[j+o][i] = \begin{cases} \sin(jl) & (i = 0) \\ \sin(jl') & (i = 1) \\ \sin(jF) & (i = 2) \\ \sin(jD) & (i = 3) \end{cases}.$$

Für l ($i=0$) erhält man etwa mittels

```
const int o=6; // Index Offset
Cos[o][i]=1.0; Cos[o+1][i]=cos(arg); Cos[o-1][i]=-Cos[o+1][i];
Sin[o][i]=0.0; Sin[o+1][i]=sin(arg); Sin[o-1][i]=-Sin[o+1][i];
for (int j=2;j<=max;j++) {
    AddThe ( Cos[o+j-1][i],Sin[o+j-1][i], Cos[o+1][i],Sin[o+1][i],
              Cos[o+j][i],Sin[o+j][i] );
    Cos[o-j][i]=-Cos[o+j][i];
    Sin[o-j][i]=-Sin[o+j][i];
}
```

alle Werte $\cos(jl)$ und $\sin(jl)$ für $j=-\text{max} \dots +\text{max}$ mit nur zwei Aufrufen der Sinus- oder Cosinusfunktion. Für die C++ Implementierung der ILE Reihenentwicklungen bietet sich die Definition einer eigenen Klasse ILE_Pert an, die unter anderem die Elemente Cos und Sin enthält.

```

// Konstanten
const int o    = 6;           // Index-Versatz
const int dim = 2*o+1;        // Dimension eines Arbeits-Feldes
// Definition der Klasse ILE_Pert zum Aufsummieren von Brown's Reihen-
// entwicklungen der Stoerungen der Mondbahn
class ILE_Pert
{
public:
    // Initialisierung (mittlere Argumente, langperiodische Korrekturen etc.)
    void Init (double T);
    // Stoerungsterm auswerten
    void Term (int p, int q, int r, int s, double& x, double& y);
    // Summation solarer Stoerungen
    void AddSol ( double coeffl, double coeffS, double coeffg,
                  double coeffP, int p, int q, int r, int s );
    // Summation der Stoerungen in Breite
    void AddN (double coeffN, int p, int q, int r, int s);
    // Planetare Stoerungen
    void Planetary (double T);
    // Koordinaten
    double lambda(); // ekliptikale Laenge in [rad]
    double beta();   // ekliptikale Breite in [rad]
    double dist();   // geozentrische Entfernung in [km]
private:
    double Dgam;           // Langperiodische Stoerung
    double Dlam, DS, gam1C, sinPi, N; // Periodische Stoerungen
    double L0, l, ls, F, D; // Mittlere Argumente der Mondbahn
    double Cos[dim][4], Sin[dim][4]; // cos und sin d. mittleren Argumente
};

```

Die Methode `Term`, die auf die zuvor berechneten Werte `Cos[] []` und `Sin[] []` zugreift, erlaubt dann die einfache Berechnung von

$$x = \cos(pl + ql' + rF + sD) \quad \text{und} \quad y = \sin(pl + ql' + rF + sD)$$

zu gegebener Charakteristik (p, q, r, s) .

```

// Term: berechnet x=cos(p*l+q*ls+r*F+s*D) und y=sin(p*l+q*ls+r*F+s*D)
void ILE_Pert::Term (int p, int q, int r, int s, double& x, double& y)
{
    int i[4];
    i[0]=p; i[1]=q; i[2]=r; i[3]=s; x=1.0; y=0.0;
    for (int k=0; k<4; k++)
        if (i[k] != 0) AddThe(x,y,Cos[o+i[k]][k],Sin[o+i[k]][k],x,y);
}

```

Am Beispiel der Reihe für die Breitenstörung N (8.6) lässt sich die Anwendung dieser Funktion gut erläutern. Die Methode `AddN` hat die Aufgabe, alle Terme der genannten Störungsreihe in der Variablen N aufzusummen:

```

// AddN: Summation der Stoerungen in Breite
void ILE_Pert::AddN (double coeffN, int p, int q, int r, int s) {
    double x,y;
    Term(p,q,r,s,x,y); N += coeffN*y;
}

```

Mit Hilfe von `Term` wird zunächst der Wert der zugehörigen Winkelfunktion berechnet (`N` enthält nur sin-Terme), mit dem Koeffizienten multipliziert und zu `N` addiert. In jedem Aufruf von `AddN` wird ein einzelner Term bearbeitet, wobei als Parameter der Koeffizient und die charakteristischen Werte (p, q, r, s) des Terms übergeben werden. Natürlich müssen aber zuvor bereits alle Werte der Felder `Sin[] []` und `Cos[] []` bestimmt worden sein. Hierzu dient der Aufruf der Methode `Init`, die zu gegebener Zeit T alle notwendigen Hilfsgrößen berechnet. Der Codebaustein zur Berechnung der Breitenstörung N lautet damit:

```
ILE_Pert::Pert;
Pert.Init(T);
// Solare Stoerungen in Breite
Pert.AddN(-526.069, 0, 0,1,-2); Pert.AddN( -3.352, 0, 0,1,-4);
Pert.AddN( +44.297,+1, 0,1,-2); Pert.AddN( -6.000,+1, 0,1,-4);
Pert.AddN( +20.599,-1, 0,1, 0); Pert.AddN( -30.598,-1, 0,1,-2);
Pert.AddN( -24.649,-2, 0,1, 0); Pert.AddN( -2.000,-2, 0,1,-2);
Pert.AddN( -22.571, 0,+1,1,-2); Pert.AddN( +10.985, 0,-1,1,-2);
```

In ganz entsprechender Weise arbeitet die Methode `AddSol`

```
// AddSol: Summation solarer Stoerungsterme
void ILE_Pert::AddSol (
    double coeffl, double coeffS, double coeffg, double coeffP,
    int p, int q, int r, int s )
{
    double x,y;
    Term_(p,q,r,s,x,y);
    Dlam += coeffl*y; DS    += coeffS*y;
    gam1C += coeffg*x; sinPi += coeffP*x;
}
```

mit der die Reihen $\Delta\lambda$, ΔS , $\gamma_1 C$ und $\Delta \sin \Pi$ berechnet werden.

Die Behandlung der solaren Störungsterme in der bisher besprochenen Weise ist allerdings noch nicht ganz vollständig. Dies liegt daran, daß die Koeffizienten der Störungsterme genaugenommen etwas andere als die angegebenen Werte haben. Jeder Koeffizient der Reihen $\Delta\lambda$, ΔS , N , $\gamma_1 C$ und $\Delta \sin \Pi$ ist eigentlich eine Funktion verschiedener Parameter, wie zum Beispiel der Exzentrizität der Sonnenbahn, für die von Brown gewisse numerische Werte angesetzt wurden. Um den korrekten Werten dieser Parameter Rechnung zu tragen, muß jeder Term, der durch das Tupel (p, q, r, s) charakterisiert ist, mit einem Faktor

$$(1.000002208)^{|p|} \cdot (1.0 - 0.002495388(T+1))^{|q|} \cdot (1.000002708 + 139.978\Delta\gamma)^{|r|}$$

multipliziert werden. Darin ist $\Delta\gamma$ durch die (in `ILE_Pert::Init` berechnete) Größe

$$\begin{aligned}\Delta\gamma = & -0.000003332 \cdot \sin(2\pi(0.59734 - 5.37261T)) \\ & -0.000000539 \cdot \sin(2\pi(0.35498 - 5.37899T)) \\ & -0.000000064 \cdot \sin(2\pi(0.39943 - 5.37511T))\end{aligned}$$

gegeben. Man kann diese Korrekturen allerdings in sehr einfacher Weise berücksichtigen, wenn man bei der oben beschriebenen Berechnung der Felder `Cos[] []` und `Sin[] []` die Größen

$$\begin{aligned} \text{Cos}[o+1][0] \text{ und } \text{Sin}[o+1][0] & \quad \text{mit } (1.000002208), \\ \text{Cos}[o+1][1] \text{ und } \text{Sin}[o+1][1] & \quad \text{mit } (1.0 - 0.002495388(T+1)) \text{ und} \\ \text{Cos}[o+1][2] \text{ und } \text{Sin}[o+1][2] & \quad \text{mit } (1.000002708 + 139.978\Delta\gamma) \end{aligned}$$

multipliziert. Durch den fortgesetzten Aufruf von `AddThe` werden dann bereits alle diese vorberechneten Winkelfunktionswerte mit dem richtigen Korrekturfaktor versehen. So enthält beispielsweise `Sin[o+3,1]` statt $\sin(3l')$ den Wert $(1.0 - 0.002495388(T + 1))^{3!} \cdot \sin(3l')$. Mit dieser kleinen Änderung werden alle Terme automatisch und ohne großen Aufwand mit dem richtigen Faktor multipliziert.

Die Berechnung der verschiedenen Störungen des Mondes durch die Sonne ist damit abgeschlossen. Allerdings ist die Sonne nicht der einzige Körper des Sonnensystems, der den Mond auf seiner Bahn beeinflusst. In einer genauen Theorie der Mondbahn dürfen auch die planetaren Störungen nicht vernachlässigt werden. In der Funktion `MoonPos` werden deshalb die wichtigsten Beiträge von Venus und Jupiter berücksichtigt:

$$\begin{aligned} \Delta\lambda_{\text{plan}} = & +0.^{\circ}82 \sin(0.^{\circ}7736 \quad -62.^{\circ}5512T) + 0.^{\circ}31 \sin(0.^{\circ}0466 \quad -125.^{\circ}1025T) \\ & + 0.^{\circ}35 \sin(0.^{\circ}5785 \quad -25.^{\circ}1042T) + 0.^{\circ}66 \sin(0.^{\circ}4591 + 1335.^{\circ}8075T) \\ & + 0.^{\circ}64 \sin(0.^{\circ}3130 \quad -91.^{\circ}5680T) + 1.^{\circ}14 \sin(0.^{\circ}1480 + 1331.^{\circ}2898T) \\ & + 0.^{\circ}21 \sin(0.^{\circ}5918 + 1056.^{\circ}5859T) + 0.^{\circ}44 \sin(0.^{\circ}5784 + 1322.^{\circ}8595T) \\ & + 0.^{\circ}24 \sin(0.^{\circ}2275 \quad -5.^{\circ}7374T) + 0.^{\circ}28 \sin(0.^{\circ}2965 \quad +2.^{\circ}6929T) \\ & + 0.^{\circ}33 \sin(0.^{\circ}3132 \quad +6.^{\circ}3368T) \end{aligned}$$

Diese Störungen der ekliptikalnen Länge werden in `ILE_Pert::Planetary` berechnet.

Damit sind nun alle benötigten Größen bekannt. Die ekliptikale Länge des Mondes erhält man durch Addition der solaren und planetaren Störungen zur mittleren Länge:

$$\lambda = L_0 + \Delta\lambda + \Delta\lambda_{\text{plan}} .$$

L_0 ist dabei der bereits nach (8.7) korrigierte Wert der mittleren Länge. Für die ekliptikale Breite wird zunächst ΔS zum mittleren Knotenabstand F addiert:

$$S = F + \Delta S .$$

Damit gilt

$$\begin{aligned} \beta = & (1.000002708 + 139.978\Delta\gamma) \cdot \{18519.^{\circ}70 + \gamma_1 C\} \cdot \sin(S) \\ & - \{6.^{\circ}24\} \cdot \sin(3S) + N . \end{aligned}$$

Die Entfernung des Mondes wird aus dem Sinus der Horizontalparallaxe Π berechnet:

$$r = (1 / \sin \Pi) R_\oplus \quad (\text{Erdradius } R_\oplus \approx 6378.14 \text{ km}) .$$

$\sin \Pi$ hat den Wert

$$\sin \Pi = 0.999953253 \cdot (3422.^{\circ}7 + \Delta \sin \Pi) \cdot \frac{\pi}{180 \cdot 3600^{\circ}} .$$

Der Faktor

$$\frac{\pi}{180 \cdot 3600''} = \frac{1}{206264''81}$$

dient dabei nur zur Umrechnung von Bogensekunden ins Bogenmaß.

Es folgt nun die vollständige Funktion `MoonPos` einschließlich der noch nicht aufgelisteten Methoden der Klasse `ILE_Pert`. Zu einer gegebenen Zeit T (ausgedrückt in julianischen Jahrhunderten Ephemeridenzeit seit der Epoche J2000) berechnet `MoonPos` den Vektor der geozentrischen ekliptikalnen Koordinaten des Mondes (in [km]). Die Koordinaten beziehen sich auf das mittlere Äquinoktium des Datums.

```
// Sine
double Sine (double x) { return sin(pi2*Frac(x)); }

//
// Initialisierung
//
void ILE_Pert::Init (double T)
{
    // Variables
    double dL0, dL, dls, dF, dD;                      // Langperiodische Stoerungen
    double T2, arg, fac;                                // Hilfsgroessen
    double S1, S2, S3, S4, S5, S6, S7;
    int     max;

    T2=T*T; // Zeit
    // Stoerungen auf Null setzen
    Dlam=0.0; DS=0.0; gam1C=0.0; sinPi=3422.7000; N=0.0;

    // Langperiodische Stoerungen
    S1 = Sine (0.19833+0.05611*T); S2 = Sine (0.27869+0.04508*T);
    S3 = Sine (0.16827-0.36903*T); S4 = Sine (0.34734-5.37261*T);
    S5 = Sine (0.10498-5.37899*T); S6 = Sine (0.42681-0.41855*T);
    S7 = Sine (0.14943-5.37511*T);
    dL0 = 0.84*S1+0.31*S2+14.27*S3+ 7.26*S4+ 0.28*S5+0.24*S6;
    dL = 2.94*S1+0.31*S2+14.27*S3+ 9.34*S4+ 1.12*S5+0.83*S6;
    dls = -6.40*S1 -1.89*S6;
    dF = 0.21*S1+0.31*S2+14.27*S3-88.70*S4-15.30*S5+0.24*S6-1.86*S7;
    dD = dL0-dls;
    Dgam = -3332e-9 * Sine (0.59734-5.37261*T)
           -539e-9 * Sine (0.35498-5.37899*T)
           -64e-9 * Sine (0.39943-5.37511*T);

    // Mittlere Argumente der Mondbahn (inkl. langperiodische Korrekturen)
    // L0 mittlere Laenge des Mondes
    // l mittlere Anomalie des Mondes   l' mittlere Anomalie der Sonne
    // F mittlerer Knotenabstand      D mittlere Elongation von der Sonne
    L0 = pi2*Frac(0.60643382+1336.85522467*T-0.00000313*T2) + dL0/Arcs;
    l = pi2*Frac(0.37489701+1325.55240982*T+0.00002565*T2) + dL /Arcs;
    ls = pi2*Frac(0.99312619+ 99.99735956*T-0.00000044*T2) + dls/Arcs;
    F = pi2*Frac(0.25909118+1342.22782980*T-0.00000892*T2) + dF /Arcs;
    D = pi2*Frac(0.82736186+1236.85308708*T-0.00000397*T2) + dD /Arcs;
```

```

// Kosinus und Sinus von Vielfachen der mittleren Argumente
// einschliesslich saekularer Korrekturen
for (int i=0; i<=3; i++) {
    switch(i) {
        case 0: arg=l; max=4; fac=1.000002208; break;
        case 1: arg=ls; max=3; fac=0.997504612-0.002495388*T; break;
        case 2: arg=F; max=4; fac=1.000002708+139.978*Dgam; break;
        case 3: arg=D; max=6; fac=1.0; break;
    };
    Cos[o][i]=1.0; Cos[o+1][i]=cos(arg)*fac; Cos[o-1][i]=-Cos[o+1][i];
    Sin[o][i]=0.0; Sin[o+1][i]=sin(arg)*fac; Sin[o-1][i]=-Sin[o+1][i];
    for (int j=2;j<=max;j++) {
        AddThe ( Cos[o+j-1][i],Sin[o+j-1][i], Cos[o+1][i],Sin[o+1][i],
                  Cos[o+j][i],Sin[o+j][i] );
        Cos[o-j][i]=-Cos[o+j][i];
        Sin[o-j][i]=-Sin[o+j][i];
    };
}
};

// Planetary: Stoerungen der ekliptikalnen Breite durch Venus und Jupiter
void ILE_Pert::Planetary (double T) {
    Dlam +=
        +0.82*Sine(0.7736 -62.5512*T)+0.31*Sine(0.0466 -125.1025*T)
        +0.35*Sine(0.5785 -25.1042*T)+0.66*Sine(0.4591+1335.8075*T)
        +0.64*Sine(0.3130 -91.5680*T)+1.14*Sine(0.1480+1331.2898*T)
        +0.21*Sine(0.5918+1056.5859*T)+0.44*Sine(0.5784+1322.8595*T)
        +0.24*Sine(0.2275 -5.7374*T)+0.28*Sine(0.2965 +2.6929*T)
        +0.33*Sine(0.3132 +6.3368*T);
}

// lambda, beta, dist
double ILE_Pert::lambda() {
    return Modulo ( L0+Dlam/Arcs, pi2 );
}
double ILE_Pert::beta() {
    double S = F + DS/Arcs;
    double fac = 1.000002708+139.978*Dgam;
    return (fac*(18518.511+1.189+gam1C)*sin(S)-6.24*sin(3*S)+N) / Arcs;
}
double ILE_Pert::dist() {
    return R_Earth * Arcs / (sinPi * 0.9999953253);
}

//-----
// MoonPos: Berechnet die ekliptikale Position des Mondes
//          mit der Mondtheorie von Brown (Improved Lunar Ephemeris)
//          T          Zeit in Julianischen Jahrhunderten seit J2000
//          <return> Geozentrische Position des Mondes (in [km]) bezogen auf
//                      Ekliptik und Fruehlingspunkt des Datums
//-----
Vec3D MoonPos (double T)
{
    ILE_Pert Pert;

```

```

Pert.Init(T); // Initialisierung

// Solare Stoerungen
Pert.AddSol ( 13.902, 14.06,-0.001, 0.2607,0, 0, 0, 4);
Pert.AddSol ( 0.403, -4.01,+0.394, 0.0023,0, 0, 0, 3);
Pert.AddSol ( 2369.912, 2373.36,+0.601, 28.2333,0, 0, 0, 2);
Pert.AddSol ( -125.154, -112.79,-0.725, -0.9781,0, 0, 0, 1);
Pert.AddSol ( 1.979, 6.98,-0.445, 0.0433,1, 0, 0, 4);
Pert.AddSol ( 191.953, 192.72,+0.029, 3.0861,1, 0, 0, 2);
Pert.AddSol ( -8.466, -13.51,+0.455, -0.1093,1, 0, 0, 1);
Pert.AddSol ( 22639.500, 22609.07,+0.079, 186.5398,1, 0, 0, 0);
Pert.AddSol ( 18.609, 3.59,-0.094, 0.0118,1, 0, 0,-1);
Pert.AddSol (-4586.465,-4578.13,-0.077, 34.3117,1, 0, 0,-2);
Pert.AddSol ( +3.215, 5.44,+0.192, -0.0386,1, 0, 0,-3);
Pert.AddSol ( -38.428, -38.64,+0.001, 0.6008,1, 0, 0,-4);
Pert.AddSol ( -0.393, -1.43,-0.092, 0.0086,1, 0, 0,-6);
Pert.AddSol ( -0.289, -1.59,+0.123, -0.0053,0, 1, 0, 4);
Pert.AddSol ( -24.420, -25.10,+0.040, -0.3000,0, 1, 0, 2);
Pert.AddSol ( 18.023, 17.93,+0.007, 0.1494,0, 1, 0, 1);
Pert.AddSol ( -668.146, -126.98,-1.302, -0.3997,0, 1, 0, 0);
Pert.AddSol ( 0.560, 0.32,-0.001, -0.0037,0, 1, 0,-1);
Pert.AddSol ( -165.145, -165.06,+0.054, 1.9178,0, 1, 0,-2);
Pert.AddSol ( -1.877, -6.46,-0.416, 0.0339,0, 1, 0,-4);
Pert.AddSol ( 0.213, 1.02,-0.074, 0.0054,2, 0, 0, 4);
Pert.AddSol ( 14.387, 14.78,-0.017, 0.2833,2, 0, 0, 2);
Pert.AddSol ( -0.586, -1.20,+0.054, -0.0100,2, 0, 0, 1);
Pert.AddSol ( 769.016, 767.96,+0.107, 10.1657,2, 0, 0, 0);
Pert.AddSol ( +1.750, 2.01,-0.018, 0.0155,2, 0, 0,-1);
Pert.AddSol ( -211.656, -152.53,+5.679, -0.3039,2, 0, 0,-2);
Pert.AddSol ( +1.225, 0.91,-0.030, -0.0088,2, 0, 0,-3);
Pert.AddSol ( -30.773, -34.07,-0.308, 0.3722,2, 0, 0,-4);
Pert.AddSol ( -0.570, -1.40,-0.074, 0.0109,2, 0, 0,-6);
Pert.AddSol ( -2.921, -11.75,+0.787, -0.0484,1, 1, 0, 2);
Pert.AddSol ( +1.267, 1.52,-0.022, 0.0164,1, 1, 0, 1);
Pert.AddSol ( -109.673, -115.18,+0.461, -0.9490,1, 1, 0, 0);
Pert.AddSol ( -205.962, -182.36,+2.056, +1.4437,1, 1, 0,-2);
Pert.AddSol ( 0.233, 0.36, 0.012, -0.0025,1, 1, 0,-3);
Pert.AddSol ( -4.391, -9.66,-0.471, 0.0673,1, 1, 0,-4);
Pert.AddSol ( 0.283, 1.53,-0.111, +0.0060,1,-1, 0,+4);
Pert.AddSol ( 14.577, 31.70,-1.540, +0.2302,1,-1, 0, 2);
Pert.AddSol ( 147.687, 138.76,+0.679, +1.1528,1,-1, 0, 0);
Pert.AddSol ( -1.089, 0.55,+0.021, 0.0 ,1,-1, 0,-1);
Pert.AddSol ( 28.475, 23.59,-0.443, -0.2257,1,-1, 0,-2);
Pert.AddSol ( -0.276, -0.38,-0.006, -0.0036,1,-1, 0,-3);
Pert.AddSol ( 0.636, 2.27,+0.146, -0.0102,1,-1, 0,-4);
Pert.AddSol ( -0.189, -1.68,+0.131, -0.0028,0, 2, 0, 2);
Pert.AddSol ( -7.486, -0.66,-0.037, -0.0086,0, 2, 0, 0);
Pert.AddSol ( -8.096, -16.35,-0.740, 0.0918,0, 2, 0,-2);
Pert.AddSol ( -5.741, -0.04, 0.0 , -0.0009,0, 0, 2, 2);
Pert.AddSol ( 0.255, 0.0 , 0.0 , 0.0 ,0, 0, 2, 1);
Pert.AddSol ( -411.608, -0.20, 0.0 , -0.0124,0, 0, 2, 0);
Pert.AddSol ( 0.584, 0.84, 0.0 , +0.0071,0, 0, 2,-1);
Pert.AddSol ( -55.173, -52.14, 0.0 , -0.1052,0, 0, 2,-2);
Pert.AddSol ( 0.254, 0.25, 0.0 , -0.0017,0, 0, 2,-3);
Pert.AddSol ( +0.025, -1.67, 0.0 , +0.0031,0, 0, 2,-4);

```

```

Pert.AddSol ( 1.060,    2.96,-0.166,   0.0243,3, 0, 0,+2);
Pert.AddSol ( 36.124,   50.64,-1.300,   0.6215,3, 0, 0, 0);
Pert.AddSol ( -13.193,  -16.40,+0.258,  -0.1187,3, 0, 0,-2);
Pert.AddSol ( -1.187,   -0.74,+0.042,   0.0074,3, 0, 0,-4);
Pert.AddSol ( -0.293,   -0.31,-0.002,   0.0046,3, 0, 0,-6);
Pert.AddSol ( -0.290,   -1.45,+0.116,  -0.0051,2, 1, 0, 2);
Pert.AddSol ( -7.649,   -10.56,+0.259,  -0.1038,2, 1, 0, 0);
Pert.AddSol ( -8.627,   -7.55,+0.078,  -0.0192,2, 1, 0,-2);
Pert.AddSol ( -2.740,   -2.54,+0.022,   0.0324,2, 1, 0,-4);
Pert.AddSol ( 1.181,    3.32,-0.212,   0.0213,2,-1, 0,+2);
Pert.AddSol ( 9.703,    11.67,-0.151,   0.1268,2,-1, 0, 0);
Pert.AddSol ( -0.352,   -0.37,+0.001,  -0.0028,2,-1, 0,-1);
Pert.AddSol ( -2.494,   -1.17,-0.003,  -0.0017,2,-1, 0,-2);
Pert.AddSol ( 0.360,    0.20,-0.012,  -0.0043,2,-1, 0,-4);
Pert.AddSol ( -1.167,   -1.25,+0.008,  -0.0106,1, 2, 0, 0);
Pert.AddSol ( -7.412,   -6.12,+0.117,   0.0484,1, 2, 0,-2);
Pert.AddSol ( -0.311,   -0.65,-0.032,   0.0044,1, 2, 0,-4);
Pert.AddSol ( +0.757,   1.82,-0.105,   0.0112,1,-2, 0, 2);
Pert.AddSol ( +2.580,   2.32,+0.027,   0.0196,1,-2, 0, 0);
Pert.AddSol ( +2.533,   2.40,-0.014,  -0.0212,1,-2, 0,-2);
Pert.AddSol ( -0.344,   -0.57,-0.025,  +0.0036,0, 3, 0,-2);
Pert.AddSol ( -0.992,   -0.02, 0.0 ,   0.0 , 1, 0, 2, 2);
Pert.AddSol ( -45.099,  -0.02, 0.0 ,   -0.0010,1, 0, 2, 0);
Pert.AddSol ( -0.179,   -9.52, 0.0 ,   -0.0833,1, 0, 2,-2);
Pert.AddSol ( -0.301,   -0.33, 0.0 ,   0.0014,1, 0, 2,-4);
Pert.AddSol ( -6.382,   -3.37, 0.0 ,   -0.0481,1, 0,-2, 2);
Pert.AddSol ( 39.528,   85.13, 0.0 ,   -0.7136,1, 0,-2, 0);
Pert.AddSol ( 9.366,    0.71, 0.0 ,   -0.0112,1, 0,-2,-2);
Pert.AddSol ( 0.202,    0.02, 0.0 ,   0.0 , 1, 0,-2,-4);
Pert.AddSol ( 0.415,    0.10, 0.0 ,   0.0013,0, 1, 2, 0);
Pert.AddSol ( -2.152,   -2.26, 0.0 ,   -0.0066,0, 1, 2,-2);
Pert.AddSol ( -1.440,   -1.30, 0.0 ,   +0.0014,0, 1,-2, 2);
Pert.AddSol ( 0.384,    -0.04, 0.0 ,   0.0 , 0, 1,-2,-2);
Pert.AddSol ( +1.938,   +3.60,-0.145,  +0.0401,4, 0, 0, 0);
Pert.AddSol ( -0.952,   -1.58,+0.052,  -0.0130,4, 0, 0,-2);
Pert.AddSol ( -0.551,   -0.94,+0.032,  -0.0097,3, 1, 0, 0);
Pert.AddSol ( -0.482,   -0.57,+0.005,  -0.0045,3, 1, 0,-2);
Pert.AddSol ( 0.681,    0.96,-0.026,  0.0115,3,-1, 0, 0);
Pert.AddSol ( -0.297,   -0.27, 0.002,  -0.0009,2, 2, 0,-2);
Pert.AddSol ( 0.254,    +0.21,-0.003,  0.0 , 2,-2, 0,-2);
Pert.AddSol ( -0.250,   -0.22, 0.004,  0.0014,1, 3, 0,-2);
Pert.AddSol ( -3.996,   0.0 , 0.0 ,  +0.0004,2, 0, 2, 0);
Pert.AddSol ( 0.557,    -0.75, 0.0 ,  -0.0090,2, 0, 2,-2);
Pert.AddSol ( -0.459,   -0.38, 0.0 ,  -0.0053,2, 0,-2, 2);
Pert.AddSol ( -1.298,   0.74, 0.0 ,  +0.0004,2, 0,-2, 0);
Pert.AddSol ( 0.538,    1.14, 0.0 ,  -0.0141,2, 0,-2,-2);
Pert.AddSol ( 0.263,    0.02, 0.0 ,  0.0 , 1, 1, 2, 0);
Pert.AddSol ( 0.426,    +0.07, 0.0 ,  -0.0006,1, 1,-2,-2);
Pert.AddSol ( -0.304,   +0.03, 0.0 ,  +0.0003,1,-1, 2, 0);
Pert.AddSol ( -0.372,   -0.19, 0.0 ,  -0.0027,1,-1,-2, 2);
Pert.AddSol ( +0.418,   0.0 , 0.0 ,  0.0 , 0, 0, 4, 0);
Pert.AddSol ( -0.330,   -0.04, 0.0 ,  0.0 , 3, 0, 2, 0);

// Solare Stoerungen in Breite
Pert.AddN(-526.069, 0, 0,1,-2);  Pert.AddN( -3.352, 0, 0,1,-4);

```

```

Pert.AddN( +44.297,+1, 0,1,-2);   Pert.AddN( -6.000,+1, 0,1,-4);
Pert.AddN( +20.599,-1, 0,1, 0);   Pert.AddN( -30.598,-1, 0,1,-2);
Pert.AddN( -24.649,-2, 0,1, 0);   Pert.AddN( -2.000,-2, 0,1,-2);
Pert.AddN( -22.571, 0,+1,1,-2);   Pert.AddN( +10.985, 0,-1,1,-2);

// Planetare Stoerungen
Pert.Planetary(T);

// Position
return Vec3D ( Polar(Pert.lambda(),Pert.beta(),Pert.dist()) );
}

```

Die Funktion `MoonPos` wird noch durch die Funktion `MoonEqu` ergänzt, die anstelle der mittleren ekliptikalnen Koordinaten die wahren äquatorialen Koordinaten berechnet.

```

//-----
// MoonEqu: Berechnet die aequatoriale Position des Mondes
//           mit der Mondtheorie von Brown (Improved Lunar Ephemeris)
//   T          Zeit in Julianischen Jahrhunderten seit J2000
// <return>    Geozentrische Position des Mondes (in [km]) bezogen auf
//             Aequator und Fruehlingspunkt des Datums
//-----
Vec3D MoonEqu (double T)
{
    return NutMatrix(T) * Ecl2EquMatrix(T) * MoonPos(T);
}

```

8.3 Tschebyscheff-Approximation

Trotz aller Kunstgriffe bei der Auswertung der Störungsreihen erfordert die Berechnung der Mondkoordinaten mit der Funktion `MoonPos` noch einen relativ hohen Aufwand. Dies macht sich besonders dort sehr unangenehm bemerkbar, wo viele Mondpositionen auf einmal benötigt werden. Zum Beispiel wollen wir im übernächsten Kapitel Sternbedeckungen durch den Mond vorhersagen. Dabei wird iterativ die Konjunktionszeit des Mondes mit einem bestimmten Stern gesucht. Um derart rechenintensive Aufgaben effizient lösen zu können, soll nun noch eine spezielle C++ Klasse entwickelt werden, mit deren Hilfe man die Koordinaten eines Himmelskörpers in praktisch geeigneter Form approximieren kann. Damit kann immer dann Rechenzeit einsparen, wenn mehr Koordinatenwerte benötigt werden, als man für die Aufstellung der Näherung braucht.

Ein einfaches Beispiel einer Approximation haben wir schon kennengelernt, als wir in Kap. 3 den Höhenverlauf von Sonne und Mond abschnittsweise durch Parabeln beschrieben haben. Hier wurde zu je drei Stützstellen der Funktion ein Polynom zweiten Grades bestimmt, das dann den Funktionsverlauf in einem Intervall angenähert darstellt. Ganz allgemein kann man zu n voneinander verschiedenen Punkten (x_i, y_i) mit $i = 1 \dots n$ in eindeutiger Weise ein Polynom $P_n(x)$ vom Grad $n - 1$ bestimmen, das genau durch die vorgegebenen Punkte verläuft. Um dieses Polynom zu ermitteln, gibt es eine ganze Reihe von Verfahren. Wir wollen hier nur die von Lagrange und Newton nennen.

Leider aber erweist sich ein solchermaßen ermitteltes Polynom in vielen praktischen Fällen als ungeeignet. Die vorgegebenen Funktionswerte werden zwar immer richtig wiedergegeben, zwischen den Stützstellen neigt das Polynom aber zu Schwingungen, die im allgemeinen nichts mit dem Verlauf der approximierten Funktion zu tun haben und zum Rand des Intervalls unkontrolliert anwachsen. Dieser Effekt ist um so stärker, je höher man den Grad des Polynoms wählt. Brauchbare Resultate erhält man deshalb mit der Lagrange-Interpolation meist nur bei niedrigen Ordnungen (etwa bis $n = 5$). Wir benötigen aber ein Verfahren, bei dem eine gleichmäßige Approximation auch bei hohen Ordnungen der Polynome gewährleistet ist. Im folgenden soll gezeigt werden, wie man eine solche Näherung findet. Den Schlüssel dazu bilden die sogenannten *Tschebyscheff-Polynome* (Abb. 8.1).

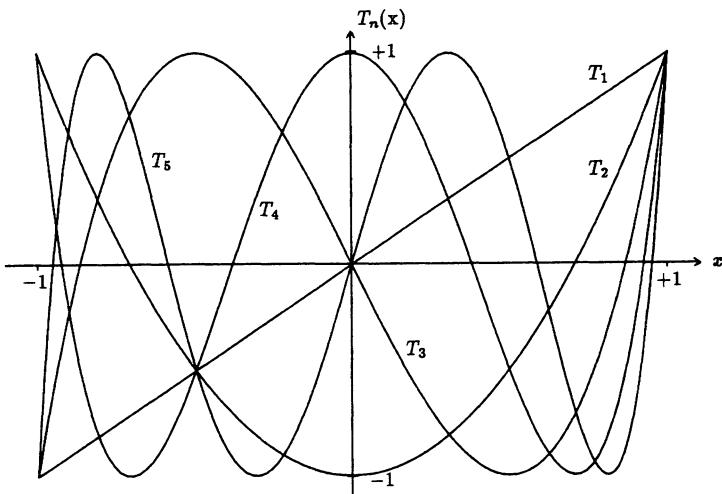


Abb. 8.1. Die Tschebyscheff-Polynome T_1 bis T_5

Das Tschebyscheff-Polynom n -ten Grades ist (für $|x| \leq 1$) als

$$T_n(x) = \cos(n \cdot \arccos x) \quad (8.9)$$

definiert. Dieser trigonometrischen Darstellung sieht man zunächst nicht an, daß es sich in Wirklichkeit um Polynome handelt. Nur für $n = 0$ ($T_0 = 1$) und $n = 1$ ($T_1 = x$) ist dies noch leicht zu erkennen. Aus dem Additionstheorem für die Cosinus-Funktion kann man aber sehr schnell eine Rekursionsbeziehung für die T_n ableiten, aus der man unmittelbar sieht, daß der Ausdruck (8.9) tatsächlich Polynome definiert. Es gilt nämlich allgemein

$$\cos(\alpha + \beta) + \cos(\alpha - \beta) = 2 \cos(\alpha) \cos(\beta)$$

und damit auch

$$\cos((n+1)\varphi) = 2 \cos(n\varphi) \cos(\varphi) - \cos((n-1)\varphi) . \quad (8.10)$$

Setzt man hierin $\varphi = \arccos x$, dann folgt

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad \text{für } n \geq 1 \quad . \quad (8.11)$$

Von dieser Rekursionsbeziehung werden wir später ausführlich Gebrauch machen. $T_n(x)$ ist aufgrund dieses Zusammenhangs jedenfalls sicher ein Polynom vom Grad n in x . Ausgeschrieben lauten die ersten Tschebyscheff-Polynome:

$$\begin{aligned} T_0(x) &= 1 & T_3(x) &= 4x^3 - 3x \\ T_1(x) &= x & T_4(x) &= 8x^4 - 8x^2 + 1 \\ T_2(x) &= 2x^2 - 1 & T_5(x) &= 16x^5 - 20x^3 + 5x \end{aligned} \quad .$$

Wir wollen jetzt die Eigenschaften der Tschebyscheff-Polynome im Intervall $[-1, +1]$ genauer betrachten. Wie man durch Einsetzen in die Definition (8.9) leicht erkennen kann, hat $T_n(x)$ in diesem Bereich genau n Nullstellen:

$$T_n(x) = 0 \quad \text{für } x = \cos(\pi \cdot (k + 1/2)/n) \quad \text{mit } k = 0, 1, \dots, n-1 \quad . \quad (8.12)$$

Die Funktionswerte von $T_n(x)$ bleiben für $|x| \leq 1$ betragsmäßig auf 1 beschränkt.

Um eine beliebige im Intervall $[a, b]$ definierte Funktion $f(x)$ zu approximieren, ersetzen wir zunächst die unabhängige Variable x durch die normalisierte Variable \hat{x} aus dem Intervall $[-1, +1]$. Dazu benutzen wir die Umrechnungen

$$\hat{x} = \frac{x - \frac{1}{2}(a + b)}{\frac{1}{2}(b - a)} \quad \text{für } x \in [a, b] \rightarrow \hat{x} \in [-1, +1]$$

und

$$x = \hat{x} \cdot \frac{1}{2}(b - a) + \frac{1}{2}(a + b) \quad \text{für } \hat{x} \in [-1, +1] \rightarrow x \in [a, b] \quad .$$

Eine Funktion $f(x)$ lässt sich dann in der Form

$$f(x) \approx f^*(x) = \sum_{j=0}^n c_j T_j(\hat{x}) - c_0/2 \quad (8.13)$$

durch Tschebyscheff-Polynome bis zum Grad n darstellen. Die Koeffizienten c_j dieser Summe werden aus der Beziehung

$$c_j = \frac{2}{n+1} \sum_{k=1}^{n+1} f(x_k^{n+1}) T_j(\hat{x}_k^{n+1}) \quad (8.14)$$

berechnet, wobei \hat{x}_k^{n+1} die k -te Nullstelle von T_{n+1} darstellt. Ausführlich geschrieben lauten die einzelnen Terme in dieser Beziehung wegen (8.9) und (8.12):

$$\begin{aligned} x_k^{n+1} &= \frac{(b-a)}{2} \cdot \cos\left(\pi \frac{2k-1}{2n+2}\right) + \frac{(a+b)}{2} \quad k = 0, 1, \dots, n \\ T_j(\hat{x}_k^{n+1}) &= \cos\left(j\pi \frac{2k-1}{2n+2}\right) \end{aligned} \quad .$$

Man erkennt, daß die gegebene Funktion f zur Berechnung der Koeffizienten c_j an $(n+1)$ Punkten auszuwerten ist. Diese sind als Nullstellen von T_{n+1} fest vorgegeben und nicht mehr frei wählbar wie bei der Lagrange-Interpolation. Hierin

aber liegt gerade das Geheimnis für das gutartige Verhalten der Tschebyscheff-Approximation. f^* ist nicht nur ein Polynom n -ten Grades, das mit f an $(n+1)$ Punkten ($f^*(x_k^{n+1}) = f(x_k^{n+1})$) übereinstimmt. Durch die Wahl der Stützstellen, die am Rand des Interpolationsintervalls dichter liegen als in der Mitte, wird vielmehr überall ein gleichmäßiger Approximationsfehler gewährleistet. Man erkennt dies besonders gut, wenn man bei der Berechnung von f^* nicht alle Terme in (8.13) aufsummier. Vernachlässigt man etwa das höchste Glied $c_n T_n$, dann kann man sicher sagen, daß der resultierende Fehler wegen $|T_n| \leq 1$ in ganz $[a, b]$ kleiner als $|c_n|$ ist. Eine derartige Abschätzung ist bei anderen Interpolationsverfahren nicht möglich.

Natürlich kann man die Tschebyscheff-Approximation in dieser Form nur verwenden, wenn man auch die jeweils benötigten Funktionswerte an allen vorgegebenen Stellen berechnen kann. Wir wollen aber die Koordinaten von Himmelskörpern entwickeln, die wir im Prinzip für beliebige Zeitpunkte kennen. Deswegen spielt diese Einschränkung für unsere Anwendung keine Rolle.

Allgemein gehen wir im folgenden davon aus, daß eine zeitabhängige Funktion gegeben ist, die als Ergebnis einen dreidimensionalen Vektor des Typs `Vec3D` liefert:

```
// Prototyp der zu approximierenden Funktion: Vec3D f(double t)
typedef Vec3D (*C3Dfunct)(double t);
```

Beispiele hierzu sind die Funktionen `MoonPos` und `MoonEqu`, aber auch die in Kap. 6 beschriebenen Funktionen zur Berechnung heliozentrischer Planetenpositionen. Für Funktionen des genannten Typs können mit Hilfe der Klasse `Cheb3D` Tschebyscheff-Approximationen berechnet und ausgewertet werden.

```
// Cheb3D: Tschebyscheff-Approximation in drei Dimensionen
class Cheb3D
{
public:
    // Konstruktor, Destruktor
    Cheb3D (
        C3Dfunct f,    // Zeiger auf die zu approximierende Funktion
        int n,          // Ordnung der Approximationspolynome
        double dt       // Intervallbreite
    );
    ~Cheb3D();
    // Fit: Approximiert die Funktion f auf dem Intervall [ta,tb]
    void Fit (double ta, double tb);
    // Value: Auswertung der Approximation fuer das Argument t.
    //         Bei Bedarf wird eine neue Approximation durchgefuehrt.
    Vec3D Value (double t);
private:
    C3Dfunct m_f;           // Funktion
    int     m_n;             // Grad
    bool    m_Valid;          // Koeffizienten gueltig?
    double  m_dt;             // Intervallbreite
    double  m_ta,m_tb;        // Intervallgrenzen
    double *Cx,*Cy,*Cz;      // Koeffizienten der Approximationspolynome
};
```

Der Konstruktor

```
Cheb3D::Cheb3D (C3Dfunct f, int n, double dt)
: m_n(n), m_f(f), m_dt(dt), m_Valid(false)
{
    Cx = new double[n+1]; Cy = new double[n+1]; Cz = new double[n+1];
};
```

übernimmt dabei zunächst die Aufgabe, den notwendigen Speicherplatz für die Polynomkoeffizienten C_x , C_y und C_z der x -, y - und z -Koordinate anzulegen. Zusätzlich werden der Grad der Polynome, die gewünschte Standard-Intervalllänge und ein Zeiger auf die zu approximierende Funktion in entsprechenden Attributen gespeichert. Anschließend können mit Hilfe der Methode `Fit` die Koeffizienten der Tschebyscheff-Polynome für alle drei Koordinaten über einem vorgegebenen Intervall $[t_a, t_b]$ berechnet werden.

```
// Fit: Approximiert die Funktion m_f auf dem Intervall [ta,tb]
void Cheb3D::Fit (double ta, double tb)
{
    int      j, k;
    double   tau, t, fac;
    double* T=new double[m_n+1];
    Vec3D   f;
    // Koeffizienten loeschen
    for (j=0; j<=m_n; j++) Cx[j]=Cy[j]=Cz[j]=0.0;
    // Schleife ueber die Nullstellen von T^(n+1)
    for (k=0; k<=m_n; k++) {
        tau = cos((2*k+1)*pi/(2*m_n+2));           // Nullstelle tau_k
        t = ((tb-ta)/2.0) * tau + ((tb+ta)/2.0); // Argument t
        f = m_f(t);                                // Funktionswert f(t)
        // Rekursive Berechnung der Koeffizienten C_j
        for (j=0; j<=m_n; j++) {
            switch (j) {
                case 0: T[j]=1.0; break;
                case 1: T[j]=tau; break;
                default: T[j]=2.0*tau*T[j-1]-T[j-2];
            };
            // Koeffizient C_j um f(t)*T_j(tau) inkrementieren
            Cx[j]+=f[x]*T[j]; Cy[j]+=f[y]*T[j]; Cz[j]+=f[z]*T[j];
        };
    };
    // Skalierung
    fac = 2.0/(m_n+1);
    for (j=0; j<=m_n; j++) { Cx[j]*=fac; Cy[j]*=fac; Cz[j]*=fac; }
    // Daten aktualisieren
    m_ta = ta;
    m_tb = tb;
    m_Valid = true;
    delete[] T;
}
```

Zu den einzelnen Nullstellen $\tau = \hat{x}_k^{n+1}$ des Tschebyscheffpolynoms der Ordnung $n+1$ werden darin die Funktionswerte $f(t)$ und – rekursiv – die Werte der $T_j(\tau)$ der

Tschebyscheffpolynome der Ordnungen $j = 0, \dots, n$ bestimmt. Durch Summation ergeben sich hieraus gemäß (8.14) die gesuchten Koeffizienten.

Bei der Auswertung einer gegebenen Entwicklung nach Tschebyscheff-Polynomen ist es nicht nötig, die Polynome auch explizit zu berechnen. Ein von Clenshaw angegebener Algorithmus liefert unter Anwendung der Rekursionsbeziehung aus (8.11) folgende Vorschrift zur Auswertung einer Tschebyscheff-Entwicklung (8.13) vom Grad n mit den Koeffizienten (c_0, c_1, \dots, c_n) :

- Setze $f_{n+1} = 0$ und $f_{n+2} = 0$.
- Berechne mit dem normalisierten Argument \hat{x} die Folge

$$f_i = 2\hat{x}f_{i+1} - f_{i+2} + c_i \quad \text{für } i = n, n-1, \dots, 0 .$$

- Der gesuchte Funktionswert ist dann

$$f(x) = (f_0 - f_2)/2 = \hat{x}f_1 - f_2 + c_0/2 .$$

Die Methode `Value` wertet eine Tschebyscheff-Entwicklung nach diesem Schema aus:

```
// Value: Auswertung der Approximation fuer das Argument t
Vec3D Cheb3D::Value (double t)
{
    const double eps=0.01; // Relative Ueberlappung
    Vec3D f1, f2, old_f1;
    double tau, k;
    // Bei Bedarf Koeffizienten neu berechnen
    if ( !_m_Valid || (t<m_ta) || (m_tb<t) ) {
        k = floor(t/m_dt);
        Fit ( (k-eps)*m_dt, (k+1+eps)*m_dt );
    }
    // Approximation auswerten
    tau = (2.0*t-m_ta-m_tb)/(m_tb-m_ta);
    for (int i=m_n; i>=1; i--) {
        old_f1 = f1;
        f1 = 2.0*tau*f1-f2+Vec3D(Cx[i],Cy[i],Cz[i]);
        f2 = old_f1;
    };
    return tau*f1-f2+0.5*Vec3D(Cx[0],Cy[0],Cz[0]);
}
```

`Value` prüft dabei selbstständig, ob die gewünschte Zeit t innerhalb des Approximationsintervalls $[t_a, t_b]$ liegt. Ist dies nicht der Fall, dann wird zunächst mittels `Fit` eine Tschebyscheff-Approximation in einem geeigneten Intervall der Länge dt um t herum bestimmt. Anschließend wird diese Approximation an der Stelle t ausgewertet. Die Benutzung der Klasse `Cheb3D` wird so besonders einfach, da nach Allokation eines entsprechenden Objektes nur noch die Methode `Value` aufgerufen werden muß, während die Berechnung der Tschebyscheff-Koeffizienten automatisch im Hintergrund durchgeführt wird.

8.4 Das Programm LUNA

Das Programm Luna faßt die verschiedenen Klassen und Funktionen dieses Kapitels zu einer kleinen Anwendung zusammen. Man kann damit eine Mondephemeride – also eine Tabelle von Mondpositionen – berechnen, wie sie in vielen Jahrbüchern zu finden ist. Ausgegeben werden die scheinbaren (auf das Äquinoktium des Datums bezogenen) äquatorialen Koordinaten des Mondes, seine Entfernung in Erdradien sowie seine Äquatorial-Horizontalparallaxe. Luna entwickelt die Mondkoordinaten jeweils für einen Zeitraum von zehn Tagen in eine Reihe von Tschebyscheff-Polynomen. Diese Reihenentwicklung lässt sich anschließend einfach und schnell auswerten. Zum Aufstellen dieser Reihe mit der geforderten Approximationsgenauigkeit werden 11 Mondpositionen berechnet. Benötigt man die Mondkoordinaten also nur mit einer Schrittweite von etwa einem Tag oder mehr, so entsteht durch die Reihenentwicklung ein gewisser Mehraufwand. Ephemeriden mit kleiner Schrittweite (wie sie etwa für die Zwecke der Navigation benötigt werden) können dagegen mit dieser Technik mit beträchtlichem Zeitvorteil aufgestellt werden.

```
//-----
// Datei: Luna.cpp
// Zweck: Mondephemeride
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//-----

#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>
#include "APC_Cheb.h"
#include "APC_Const.h"
#include "APC_Math.h"
#include "APC_Moon.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"
using namespace std;

//-----
// GetEph: Eingabe des Zeitraums der Ephemeride, der Schrittweite und des
//         Äquinoktiums
// MjdStart Anfangsdatum der Ephemeride als MJD
// Step     Schrittweite der Ephemeride in [d]
// MjdEnd   Enddatum der Ephemeride als MJD
//-----
void GetEph (double& MjdStart, double& Step, double& MjdEnd)
{
    int year, month, day;
    double hour;
    cout << endl
        << " Beginn und Ende der Ephemeride: " << endl
        << endl;
    cout << " Erstes Berechnungsdatum (JJJJ MM TT HH.HHH) ... ";
    cin >> year >> month >> day >> hour; cin.ignore(81, '\n');
```

```

MjdStart = Mjd(year,month,day) + hour/24.0 ;
cout << " Letztes Berechnungsdatum (JJJJ MM TT HH.HHH) ... ";
cin >> year >> month >> day >> hour; cin.ignore(81,'\'n');
MjdEnd = Mjd(year,month,day) + hour/24.0 ;
cout << " Schrittweite (TT HH.HH) ... ";
cin >> day >> hour; cin.ignore(81,'\'n');
Step = day + hour/24.0;
}

//-----
// Hauptprogramm
// -----
void main()
{
    const int Degree = 10;
    const double Interval = 10.0/36525.0; // 10d in Jahrhunderten
    int n_line = 0;
    double MjdStart, Step, MjdEnd;
    double Date, T;
    double Dist, Parallax;
    Vec3D r_Moon;
    Cheb3D ChebMoonEqu (MoonEqu, Degree, Interval);

    // Titel
    cout << endl
        << "                               LUNA: Mondephemeride      " << endl
        << "           (c) 1999 Oliver Montenbruck, Thomas Pfleger   " << endl
        << endl;
    // Abfrage von Anfang, Schrittweite und Ende der Ephemeride
    GetEph (MjdStart, Step, MjdEnd);
    // Vorspann
    cout << endl << endl
        << "      Datum      ET          RA          Dec          Entfernung "
        << " Parallaxe " << endl
        << "                  h  m   s       o ' \\" Erdradien"
        << "      ,  \" " << endl;

    // Ephemeride berechnen
    Date = MjdStart;
    // Schleife ueber die Berechnungszeitpunkte
    while ( Date < MjdEnd + Step/2 ) {
        // Geozentrische Koordinaten des Mondes aus Tschebyscheff-Approximation
        // (Aequator und Aequinoktium des Datums)
        T = ( Date - MJD_J2000 ) / 36525.0;
        r_Moon = ChebMoonEqu.Value(T);
        // Entfernung und Parallaxe
        Dist = r_Moon[r];
        Parallax = asin(R_Earth/Dist);
        // Ausgabe
        cout << " " << DateTime(Date,HHMM)
            << fixed
            << setprecision(2) << setw(14) << Angle(Deg*r_Moon[phi]/15.0,DMMSSs)
            << " " << showpos
    }
}

```

```

<< setprecision(1) << setw(11) << Angle(Deg*r_Moon[theta],DMMSSs)
<< noshowpos
<< setprecision(3) << setw(10) << Dist/R_Earth
<< setprecision(2) << setw(12) << Angle(60.0*Deg*Parallax,DMMm)
<< endl;
++n_line; if ( (n_line % 5) ==0 ) cout << endl;
Date += Step; // Naechster Berechnungszeitpunkt
};

}
}

```

Startet man Luna, so wird zunächst die Entwicklung der Mondkoordinaten berechnet. Dann werden die Koordinaten in sehr schneller Folge ausgegeben, bis eine erneute Reihenentwicklung (für den anschließenden Zeitraum) nötig wird.

Wir wollen als Beispiel eine Mondephemeride für den Januar 1989 mit einer Schrittweite von zwei Tagen berechnen lassen, um die Bedienung des Programms zu erläutern. Luna fordert als Eingabe nur die Eckdaten der Ephemeride in der Form „Jahr, Monat, Tag und Stunde mit Dezimalbruchteil“ sowie die Schrittweite in der Form „Tag und Stunde mit Dezimalbruchteil“. Dabei ist zu beachten, daß sich hier (wie auch bei der Berechnung von Planetenpositionen) alle Zeitangaben auf die Ephemeridenzeit beziehen. Die Eingabedaten sind im folgenden durch kursive Schrift hervorgehoben.

LUNA: Mondephemeride
(c) 1999 Oliver Montenbruck, Thomas Pfleger

Beginn und Ende der Ephemeride:

Erstes Berechnungsdatum (JJJJ MM TT HH.HHH)	...	1989 01 01 00.0
Letztes Berechnungsdatum (JJJJ MM TT HH.HHH)	...	1989 01 31 00.0
Schrittweite (TT HH.HH)	...	2 00.0

Datum	ET	RA	Dec	Entfernung	Parallaxe
		h m s	o ' "	Erdradien	,
1989/01/01 00:00	13 05 26.21	-10 42 58.7	63.053	54	31.41
1989/01/03 00:00	14 38 15.31	-20 23 19.3	61.961	55	29.10
1989/01/05 00:00	16 26 11.87	-26 51 42.9	60.407	56	54.71
1989/01/07 00:00	18 27 52.77	-27 43 03.3	58.861	58	24.44
1989/01/09 00:00	20 29 55.18	-21 44 30.5	57.789	59	29.44
1989/01/11 00:00	22 21 11.90	-10 29 51.2	57.442	59	51.00
1989/01/13 00:00	0 03 44.05	+ 2 54 49.4	57.757	59	31.44
1989/01/15 00:00	1 46 21.72	+15 30 04.5	58.481	58	47.18
1989/01/17 00:00	3 36 54.09	+24 39 14.7	59.389	57	53.29
1989/01/19 00:00	5 35 18.56	+28 13 06.1	60.375	56	56.57
1989/01/21 00:00	7 31 01.72	+25 29 44.7	61.406	55	59.16
1989/01/23 00:00	9 14 03.91	+17 51 26.1	62.408	55	05.27
1989/01/25 00:00	10 44 27.57	+ 7 31 36.2	63.194	54	24.13
1989/01/27 00:00	12 08 35.18	- 3 38 59.4	63.512	54	07.79
1989/01/29 00:00	13 34 26.38	-14 17 56.9	63.141	54	26.88
1989/01/31 00:00	15 10 02.69	-23 01 46.5	62.014	55	26.26

Auffallend sind hier die hohen Deklinationen, die der Mond etwa am 7. und am 19. Januar 1989 erreicht. In diesem Jahr liegt die Knotenlinie der Mondbahn so, daß sich die Schiefe der Ekliptik (ca. $23^\circ 5'$) und die Neigung der Mondbahn gegen die Ekliptik (ca. $5^\circ 1'$) gleichsinnig überlagern und so Deklinationen von mehr als 28° auftreten können. Der Mond steht in diesem Jahr dann zu bestimmten Zeiten auffallend hoch oder tief am Himmel, und auch seine Auf- und Untergangsazimute können Extremwerte annehmen.

9. Sonnenfinsternisse

Etwa alle 30 Tage wendet der Mond zur Zeit des Neumonds der Erde seine unbelichtete Seite zu. Für einen Beobachter, der von Norden auf die Ekliptik sieht, stehen Sonne, Mond und Erde dann in einer Reihe. Dennoch trifft der Schatten, den der Mond im Sonnenlicht wirft, nur selten die Erde. Durch die Neigung seiner Bahn steht der Mond zur Neumondzeit meist etwas ober- oder unterhalb der Erdbahnebene, so daß sein Schatten die Erde verfehlt. Nur an zwei Tagen im Monat durchkreuzt der Mond die Ekliptik. Fällt einer dieser sogenannten Knotendurchgänge mit dem Neumond zusammen, dann stehen Sonne, Mond und Erde so in einer Linie, daß der Mondschatte auf einen Teil der Erde trifft. Eine solche Sonnenfinsternis findet im allgemeinen zweimal pro Jahr statt. Wer sich innerhalb des rund 100 km großen Kernschattens befindet, sieht die Sonne völlig vom Mond bedeckt. Da dies ein verhältnismäßig kleines Gebiet ist, haben nur wenige Menschen jemals die Gelegenheit, selbst eine totale Sonnenfinsternis mitzuerleben.

Würde die Mondbahnebene fest im Raum stehen, dann fänden alle Finsternisse in zwei bestimmten Monaten statt. Durch die Wanderung der Knotenlinie der Mondbahn verschieben sich die Daten der Finsternisse jedoch jedes Jahr um durchschnittlich drei Wochen. Der 18.6 Jahre dauernde Umlauf des Mondknotens spiegelt sich so direkt im Rhythmus der Sonnenfinsternisse wieder (vgl. Abb. 9.1).

9.1 Mondphasen und Finsternisse

Um einen Überblick über die möglichen Sonnenfinsternisse eines Jahres zu erhalten, kann man zunächst einmal die monatlichen Neumonddaten bestimmen und die jeweilige Stellung des Mondes zur Ekliptik untersuchen. Als Neumondzeitpunkt ist dabei der Moment definiert, in dem die ekliptikalnen Längen λ_{\odot} und λ_M von Sonne und Mond übereinstimmen, in dem also die Differenz $\lambda_M - \lambda_{\odot}$ verschwindet. Im ersten Viertel beträgt die Differenz dann 90° ($\pi/2$), bei Vollmond 180° (π) und im letzten Viertel 270° ($3\pi/2$).

Die Bestimmung der Mondphasen ist demnach gleichbedeutend mit der Aufgabe, die Nullstellen der Funktion

$$\Delta\lambda = \lambda_M - \lambda_{\odot} - k \cdot \pi/2 \quad (9.1)$$

zu bestimmen, wobei der Faktor k die jeweilige Mondphase Neumond ($k = 0$), erstes Viertel ($k = 1$), Vollmond ($k = 2$) oder letztes Viertel ($k = 3$) identifiziert. Bei der Berechnung von $\Delta\lambda$ sind gegebenenfalls Vielfache von 2π zu addieren oder zu subtrahieren, da das Ergebnis zur Nullstellenbestimmung im Intervall $[-\pi, +\pi]$ liegen muß. In der nachfolgenden Funktion `PhasesFunc` werden die Funktionen

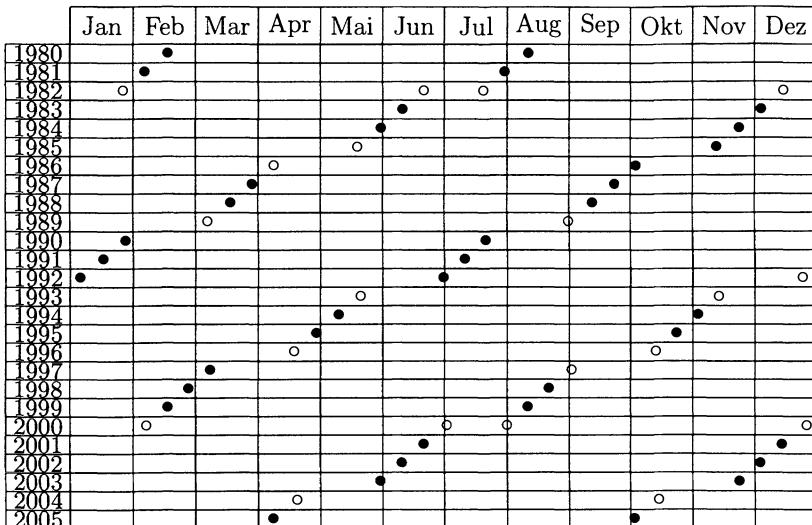


Abb. 9.1. Sonnenfinsternisse zwischen 1980 und 2005 (● =total/ringförmig, ○ =partiell)

SunPos und **MoonPos** zur Berechnung der ekliptikalnen Längen von Sonne und Mond verwendet, wobei im Falle der Sonne noch die Lichtlaufzeitkorrektur von rund acht Minuten berücksichtigt ist.

```

enum enLunarPhase { NewMoon, FirstQuarter, FullMoon, LastQuarter };
enLunarPhase Phase = NewMoon;
//-----
// PhasesFunc: Zielfunktion fuer die Suche von Mondphasen [-pi, pi]
// T          Ephemeridenzeit (in Julianischen Jahrhunderten seit J2000)
// <return>   Differenz zwischen dem Laengenabstand des Mondes von der Sonne
//             und dem dazugehoerigen Wert fuer eine gegebene Phase
//             (Neumond 0, Erstes Viertel pi/2, u.s.w.) (in [rad])
//-----
double PhasesFunc (double T)
{
    const double tau_Sun = 8.32 / (1440.0*36525.0); // 8.32 min [cy]
    const double LongDiff = MoonPos(T) [phi] - SunPos(T-tau_Sun) [phi];
    return Modulo ( LongDiff - Phase*pi/2.0 + pi, pi2 ) - pi;
}

```

Durch Suche mit einer Schrittweite von sieben Tagen lässt sich zunächst ein Zeitintervall $[t_0, t_1]$ identifizieren, innerhalb dessen ein positiver Vorzeichenwechsel von $\Delta\lambda$ stattfindet. Anschließend kann die genaue Nullstelle mit Hilfe bekannter numerischer Verfahren weiter iteriert werden. Innerhalb des Programmes **Phases** verwenden wir dazu das sogenannte Pegasus-Verfahren, das zur *regula falsi* verwandt ist, aber eine bessere Konvergenz bietet (siehe Abschn. 10.2).

Für die Identifikation möglicher Finsternisse genügt es im wesentlichen, die ekliptikale Breite β des Mondes bei Neumond (Sonnenfinsternisse) oder Vollmond

Tabelle 9.1. Kriterien des Programmes *Phases* zum Auftreten von Sonnen- und Mondfinsternissen in Abhängigkeit von der ekliptikalnen Breite β des Mondes bei Neumond oder Vollmond

Neumond		
$ \beta < 0^\circ 52'20''$	Zentrale Sonnenfinsternis sicher	c
$ \beta < 1^\circ 02'36''$	Zentrale Sonnenfinsternis möglich	c?
$ \beta < 1^\circ 24'33''$	Partielle Sonnenfinsternis sicher	p
$ \beta < 1^\circ 34'50''$	Partielle Sonnenfinsternis möglich	p?
$ \beta > 1^\circ 34'50''$	Keine Sonnenfinsternis möglich	
Vollmond		
$ \beta < 0^\circ 21'50''$	Totale Mondfinsternis sicher	t
$ \beta < 0^\circ 32'14''$	Totale Mondfinsternis möglich	t?
$ \beta < 0^\circ 53'24''$	Partielle Mondfinsternis sicher	p
$ \beta < 1^\circ 03'50''$	Partielle Mondfinsternis möglich	p?
$ \beta < 1^\circ 26'15''$	Penumbrale Mondfinsternis sicher	P
$ \beta < 1^\circ 36'43''$	Penumbrale Mondfinsternis möglich	P?
$ \beta > 1^\circ 36'43''$	Keine Mondfinsternis möglich	

(Mondfinsternisse) zu betrachten. Entsprechend der Neigung der Mondbahn gegen die Ekliptik schwankt β im Lauf des Jahres zwischen -5° und $+5^\circ$. Sonnenfinsternisse finden im allgemeinen während der beiden Neumonde mit den (betragsmäßig) kleinsten ekliptikalnen Breiten statt. In einigen Jahren kann es aber auch zu mehr als zwei Finsternissen kommen. Aktuelle Beispiele hierfür sind die Jahre 1982 und 2000 mit je vier partiellen Finsternissen. Der höchste Wert der ekliptikalnen Breite, bei dem noch eine totale Finsternis möglich ist, liegt bei etwa einem Grad. Eine partielle Sonnenfinsternis kann dagegen noch bis zu $\beta \approx 1.5^\circ$ eintreten. Zusätzlich lässt sich bereits erkennen, wo eine bestimmte Finsternis sichtbar ist. Steht der Mond etwa nördlich der Ekliptik ($\beta > 0$), dann wird das Gebiet des Schattens vornehmlich auf der Nordhalbkugel der Erde liegen.

Die behandelten Schritte sind in einem eigenen Programm *Phases* zusammengefaßt, auf dessen vollständigen Abdruck hier aber aus Platzgründen verzichtet wurde. Nach der Eingabe des gewünschten Jahres werden die Zeitpunkte der vier Mondphasen bestimmt und anhand der Kriterien in Tabelle 9.1 gekennzeichnet, zu welchen Zeitpunkten Sonnen- und Mondfinsternisse möglich oder wahrscheinlich sind. Ein Beispiel zur Anwendung des Programms ist am Ende des Kapitels aufgeführt.

9.2 Die Geometrie der Finsternis

Die Lichtstrahlen, die von der Sonne ausgehen, markieren zwei kegelförmige Gebiete, die als Kern- und Halbschatten bezeichnet werden (Abb. 9.2). Für einen Beobachter, der sich im Gebiet des Kernschattens aufhält, erscheint der Mond größer als die Sonne und bedeckt diese völlig.

In einer Entfernung von durchschnittlich 375 000 km läuft der Kernschattenkegel hinter dem Mond spitz zusammen. Daran schließt sich ein Gebiet an, in dem

die Sonnenfinsternis ringförmig erscheint. Der Mond steht von hier aus gesehen zwar als dunkle Scheibe vor der Sonne, kann diese aber nicht völlig verfinstern. Ein Teil der Sonne bleibt deshalb als heller Ring um den Mond herum sichtbar. Ein Beispiel für diesen Typ war die ringförmige Sonnenfinsternis vom 29. April 1976, die von weiten Teilen des Mittelmeeres aus beobachtet werden konnte.

Im Halbschatten (Penumbra) wird die Sonne immer nur teilweise bedeckt und erscheint als mehr oder minder breite Sichel. Der Verfinsterungsgrad ist dabei umso größer, je weiter man sich dem Kernschattenkegel nähert.

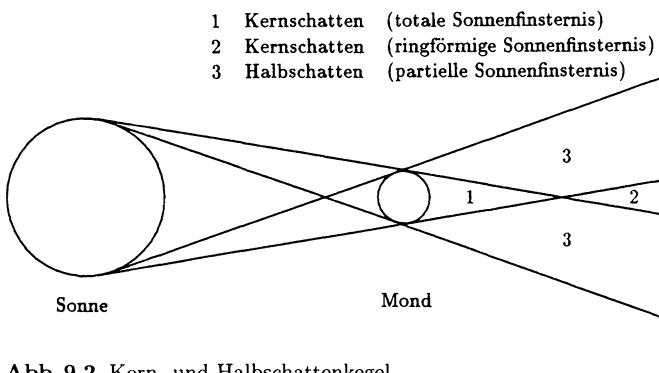


Abb. 9.2. Kern- und Halbschattenkegel

Die Entfernung des Mondes von der Erde beträgt im Mittel 380 000 km, sie schwankt aber wegen der Exzentrizität der Mondbahn um mehr als 25 000 km. Demzufolge befindet sich die Erde während einer Finsternis immer in der Nähe der Spitze des Kernschattengebietes. Selbst im Perigäum, wenn Mond und Erde sich am nächsten kommen, beträgt der Kernschattendurchmesser auf der Erdoberfläche selten mehr als 200 km. Im Apogäum verläuft die Finsternis ringförmig. Daneben gibt es als Grenzfall Finsternisse, die für einige Zeit total, ansonsten aber ringförmig sind. Ein Beispiel hierfür war die Finsternis vom 29. März 1987.

In einer Ebene, die senkrecht zur Verbindungslinie von Sonne und Mond in einer Entfernung s hinter dem Mond steht, betragen die Durchmesser von Kernschatten (d) und Halbschatten (D)

$$\begin{aligned} d(s) &= D_{\odot} \left(\frac{s}{r_{\odot M}} \right) - D_M \left(1 + \frac{s}{r_{\odot M}} \right) \quad \text{und} \\ D(s) &= D_{\odot} \left(\frac{s}{r_{\odot M}} \right) + D_M \left(1 + \frac{s}{r_{\odot M}} \right) . \end{aligned} \quad (9.2)$$

Hierin sind D_{\odot} und D_M die Durchmesser von Sonne und Mond:

$$\begin{aligned} D_{\odot} &= 1\,392\,000 \text{ km} = 218.25 R_{\oplus} \\ D_M &= 3\,476 \text{ km} = 0.5450 R_{\oplus} . \end{aligned}$$

$r_{\odot M}$ bezeichnet die Entfernung zwischen Sonne und Mond.

Die Ableitung der Formeln für den Schattendurchmesser ist in Abb. 9.3 für den Kernschatten illustriert. Dort ist f der halbe Öffnungswinkel des Schattenkegels

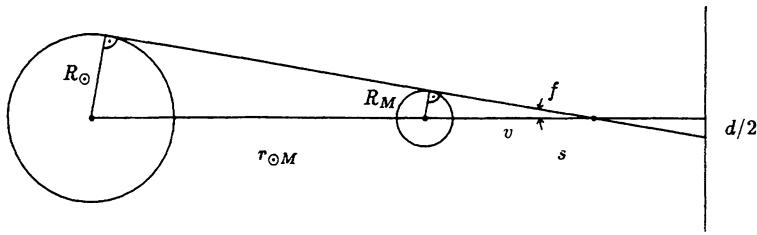


Abb. 9.3. Zur Berechnung des Schattendurchmessers

und v der Abstand der Schattenspitze vom Mondmittelpunkt. Bezeichnen R_\odot und R_M ferner die Radien von Sonne und Mond, dann gilt

$$\sin(f) \cdot v = R_M \quad \text{und} \quad \sin(f) \cdot (v + r_{\odot M}) = R_\odot$$

oder umgeformt

$$\sin(f) = \frac{R_\odot - R_M}{r_{\odot M}} \quad \text{und} \quad v = \frac{R_M \cdot r_{\odot M}}{R_\odot - R_M}.$$

Der Durchmesser des Schattens im Abstand s vom Mond ist damit

$$\begin{aligned} d &= 2(s - v) \cdot \tan(f) \\ &\approx 2(s - v) \cdot \sin(f) \\ &= 2R_\odot \left(\frac{s}{r_{\odot M}} \right) - 2R_M \left(1 + \frac{s}{r_{\odot M}} \right). \end{aligned}$$

Ganz entsprechend ist die Gleichung für den Halbschattendurchmesser D zu beweisen. Das Vorzeichen von d in den obigen Gleichungen ist so gewählt, daß d im eigentlichen Kernschattengebiet, also bei einer totalen Finsternis, *negative* und bei einer partiellen Finsternis positive Werte annimmt. Während der Halbschatten-durchmesser D in der Nähe der Erde etwa halb so groß ist wie die Erde selbst, schwankt der Durchmesser d des Kernschattens zwischen $-0.04 R_\oplus$ (250 km) und $+0.06 R_\oplus$ (350 km).

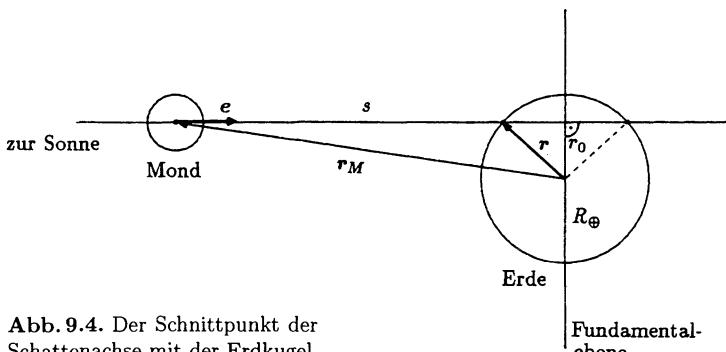


Abb. 9.4. Der Schnittpunkt der Schattenachse mit der Erdkugel

Den Ort des Kernschattens auf der Erde erhält man, wenn man eine Gerade durch die Mitte von Sonne und Mond legt und diese mit der Erdoberfläche schneidet (vgl. Abb. 9.4). Die Richtung dieser Geraden wird durch den Vektor

$$\mathbf{e} = \frac{\mathbf{r}_M - \mathbf{r}_\odot}{|\mathbf{r}_M - \mathbf{r}_\odot|} \quad (9.3)$$

der Länge Eins beschrieben, der sich aus den Koordinaten \mathbf{r}_\odot und \mathbf{r}_M von Sonne und Mond berechnen lässt. Für den Schattenpunkt \mathbf{r} gilt damit die Gleichung

$$\mathbf{r} = \mathbf{r}_M + s\mathbf{e} \quad (9.4)$$

oder, komponentenweise geschrieben,

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_M \\ y_M \\ z_M \end{pmatrix} + s \cdot \begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix} .$$

s bezeichnet den Abstand zwischen dem Schattenpunkt und der Mondmitte. Da die Spitze von \mathbf{r} auf der Erdoberfläche liegt, gilt für eine zunächst als ideale Kugel angenommene Erde mit Radius R_\oplus :

$$\mathbf{r}^2 = x^2 + y^2 + z^2 = R_\oplus^2 . \quad (9.5)$$

Zusammengenommen führt dies auf die quadratische Gleichung

$$s^2 + 2(\mathbf{r}_M \cdot \mathbf{e})s + (r_M^2 - R_\oplus^2) = 0 .$$

Sie hat die zwei Lösungen

$$s = \begin{cases} s_0 - \sqrt{\Delta} & \text{Tagseite der Erde} \\ s_0 + \sqrt{\Delta} & \text{Nachtseite der Erde} \end{cases}$$

mit

$$\begin{aligned} s_0 &= -\mathbf{r}_M \cdot \mathbf{e} = -(x_M e_x + y_M e_y + z_M e_z) \quad \text{und} \\ \Delta &= s_0^2 + R_\oplus^2 - r_M^2 . \end{aligned} \quad (9.6)$$

s_0 ist anschaulich die Entfernung des Mondes von der sogenannten Fundamental-ebene, die senkrecht auf der Schattenachse steht und durch den Erdmittelpunkt verläuft. Der gesuchte Schnittpunkt der Schattenachse mit der Erdkugel liegt im Abstand $\sqrt{\Delta}$ von dieser Ebene auf der dem Mond und der Sonne zugewandten Seite. Der zweite Schnittpunkt liegt auf der Nachtseite der Erde und hat für die Finsternis keine weitere Bedeutung. Für die Koordinaten des Kernschattens auf der Erdoberfläche ergibt sich zusammengefaßt die Lösung

$$\mathbf{r} = \mathbf{r}_M + (s_0 - \sqrt{\Delta}) \cdot \mathbf{e} . \quad (9.7)$$

Dabei wird natürlich vorausgesetzt, daß die Diskriminante Δ positiv ist, da andernfalls die Schattenachse an der Erde vorbeiläuft. Eine Sonnenfinsternis beginnt allerdings schon vor dieser zentralen Phase. Bereits wenn der Halbschattenkegel

die Erde berührt, ist von Teilen der Erde aus eine partielle Finsternis zu beobachten. Ist

$$r_0 = \sqrt{r_M^2 - s_0^2}$$

der Abstand der Schattenachse vom Erdmittelpunkt und sind d_0 und D_0 die Durchmesser von Kern- und Halbschatten auf der Fundamentalebene, dann lassen sich die einzelnen Phasen der Finsternis im wesentlichen anhand der folgenden Bedingungen unterscheiden:

$R_\oplus + D_0/2 < r_0$	keine Finsternis
$R_\oplus + d_0 /2 < r_0 < R_\oplus + D_0/2$	partielle Phase
$R_\oplus < r_0 < R_\oplus + d_0 /2$	nicht zentrale Phase
$r_0 < R_\oplus$	zentrale Phase

Während der nicht zentralen Phase streift der Kernschattenkegel die Erde, so daß von einem kleinen Gebiet aus eine totale oder ringförmige Finsternis beobachtet werden kann. Die Schattenachse selbst schneidet die Erdoberfläche in dieser Phase allerdings nicht. Es gibt deshalb keinen Ort auf der Erde, von dem aus gesehen der Mond so vor der Sonnenscheibe steht, daß deren Mittelpunkte zusammenfallen.

Die bisherigen Betrachtungen gingen davon aus, daß die Erde die Form einer idealen Kugel hat. Um auch die leichte Abplattung der Erde bei der Finsternisberechnung zu berücksichtigen, sind einige kleinere Korrekturen notwendig. Die Erdoberfläche kann man sich aus einer Kugel mit dem Radius R_\oplus entstanden denken, die parallel zur Erdachse um den Faktor

$$1 - f = 0.996647$$

gestaucht wurde. Jeder Punkt (x, y, z) dieses Rotationsellipsoids erfüllt die Gleichung

$$\mathbf{r}^2 = x^2 + y^2 + z^2/(1 - f)^2 = R_\oplus^2 , \quad (9.8)$$

die man aus (9.5) erhält, wenn man z durch $z/(1 - f)$ ersetzt. Die z -Achse des verwendeten Koordinatensystems soll dabei parallel zur Erdachse orientiert sein (äquatoriale Koordinaten).

Aus der Tatsache, daß die Erde bei einer entsprechenden Streckung in eine Kugel übergeht, ergibt sich eine einfache Methode, um den Ort des Kernschattens trotz der Abplattung richtig zu berechnen. Man multipliziert dazu die z -Koordinaten von Sonne und Mond mit dem Faktor $1/(1 - f)$, legt eine Gerade durch die so modifizierten Positionen und schneidet sie mit einer Kugel vom Radius der Erde. Der so berechnete Schnittpunkt unterscheidet sich dann lediglich durch die um den Faktor $1/(1 - f)$ zu hohe z -Koordinate vom Schnittpunkt der eigentlichen Schattenachse mit der Erdoberfläche. Bei dieser Vorgehensweise können die oben aufgestellten Beziehungen ohne Änderungen übernommen werden.

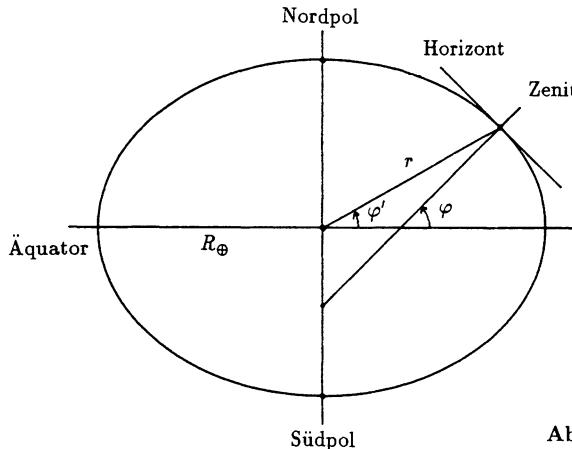


Abb. 9.5. Zur Abplattung der Erde

9.3 Geographische Koordinaten und die Abplattung der Erde

Um einen Punkt der Erdoberfläche durch Koordinaten zu kennzeichnen, verwendet man im allgemeinen die geographische Länge (λ) und die geographische Breite (φ). Diese sind eng mit den äquatorialen Koordinaten Rektaszension (α) und Deklination (δ) verwandt, mit denen in der Astronomie gearbeitet wird. Beide Koordinatensysteme sind parallel zur Äquatorebene und zur Rotationsachse der Erde ausgerichtet.

Die gegenseitige Umrechnung von Deklination und geographischer Breite würde sich erübrigen, wenn die Erde die Gestalt einer idealen Kugel hätte. In diesem Fall wären beide Koordinaten identisch. Da die Erde aber ein – wenn auch nur geringfügig – abgeplattetes Rotationsellipsoid ist, soll zunächst der Begriff der geographischen Breite etwas präzisiert werden. φ gibt an, um welchen Winkel der lokale Horizont eines Ortes gegen die Erdachse geneigt ist. Dies ist in Abb. 9.5 dargestellt. In einem Schnitt durch Nord- und Südpol hat die Erde die Form einer Ellipse, an die sich der Horizont als Tangente anschmiegt. Wie man sieht, ist φ nicht mit der geozentrischen Breite φ' beziehungsweise der Deklination δ identisch, die beide den Winkel zwischen dem Ortsvektor und der Äquatorebene angeben. φ und φ' sind über die Gleichungen

$$\begin{aligned} r \cdot \cos(\varphi') &= \frac{R_{\oplus}}{\sqrt{1 - e^2 \sin^2 \varphi}} \cos \varphi \\ r \cdot \sin(\varphi') &= \frac{(1 - e^2) R_{\oplus}}{\sqrt{1 - e^2 \sin^2 \varphi}} \sin \varphi \\ \tan(\varphi') &= (1 - e^2) \tan \varphi \end{aligned} \quad (9.9)$$

miteinander verknüpft. Hierin ist $R_{\oplus} = 6378.14$ km der Äquatorradius der Erde. e bezeichnet die Exzentrizität des Erdellipsoids. Aufgrund der bei der Rotation der Erde wirkenden Zentrifugalkräfte ist die Erde soweit verformt, daß der Äqua-

torradius rund 20 km größer ist, als die Entfernung R_{Pol} zwischen der Erdmitte und den Polen. Das Verhältnis dieser Größen definiert die Abplattung

$$f = \frac{R_{\oplus} - R_{\text{Pol}}}{R_{\oplus}} = \frac{21.385 \text{ km}}{6378.14 \text{ km}} = \frac{1}{298.257} = 0.003353 \quad . \quad (9.10)$$

Die Exzentrizität des Erdkörpers berechnet sich daraus zu

$$e = \sqrt{1 - (1 - f)^2} = \sqrt{2f - f^2} \quad . \quad (9.11)$$

Anstelle der strengen Gleichungen (9.9) kann auch die völlig ausreichende Näherung

$$\varphi = \varphi' + 0^\circ 1924 \cdot \sin(2\varphi') \quad (9.12)$$

zur Bestimmung der geographischen Breite verwendet werden. In dieser Darstellung ist gut zu erkennen, daß die Differenz zwischen φ und φ' für mittlere Breiten maximal wird und dort rund zwölf Bogenminuten beträgt. An den Polen und am Äquator verschwindet der Unterschied ganz.

Neben der geographischen Breite dient die geographische Länge als zweite Koordinate zur eindeutigen Beschreibung eines Ortes auf der Erde. Sie entspricht im wesentlichen der Rektaszension in einem äquatorialen Koordinatensystem. Während die Zählung der Rektaszension α aber am raumfesten Frühlingspunkt beginnt, wird die geographische Länge λ vom erdfesten Meridian von Greenwich aus gezählt. Die beiden Koordinaten unterscheiden sich demnach im wesentlichen um einen Winkel $\Theta_0(t)$, der über die Drehung der Erde von der Zeit t abhängt:

$$\alpha = \Theta_0(t) + \lambda \quad . \quad (9.13)$$

Für die Ortsvektoren gilt entsprechend:

$$\begin{pmatrix} r \cos(\varphi') \cos(\lambda) \\ r \cos(\varphi') \sin(\lambda) \\ r \sin(\varphi') \end{pmatrix} = \mathbf{R}_z(\Theta_0(t)) \begin{pmatrix} r \cos(\varphi') \cos(\alpha) \\ r \cos(\varphi') \sin(\alpha) \\ r \sin(\varphi') \end{pmatrix} \quad . \quad (9.14)$$

$\Theta_0(t)$ ist die jeweilige Rektaszension des Nullmeridians und damit nichts anderes als die Sternzeit von Greenwich. Die Berechnung der Sternzeit wurde bereits in Abschn. 3.3 behandelt, wo auch eine entsprechende Funktion (GMST) angegeben ist.

Eine wesentliche Schwierigkeit, die sich in diesem Zusammenhang stellt, ist die sorgfältige Unterscheidung von Weltzeit (UT) und Ephemeridenzeit (ET oder TDB/TDT). Während die Koordinaten von Sonne und Mond immer nur als Funktion der gleichförmigen Ephemeridenzeit berechnet werden können, benötigt man die Weltzeit als Argument zur Berechnung der Sternzeit. Die Weltzeit kann aber nicht mit einer Uhr gemessen werden, sondern muß gemäß ihrer Definition aus Beobachtungen ermittelt werden. Demzufolge ist es nur rückwirkend möglich, den genauen Unterschied $\Delta T = ET - UT$ zwischen beiden Zeitzählungen korrekt zu berücksichtigen. Für das zwanzigste Jahrhundert sind in Tabelle 3.1 einige gemessene Werte dieser Differenz wiedergegeben. Um die Werte nicht immer gesondert

Tabelle 9.2. Polynomapproximationen der Differenz $\Delta T = ET - UT$ von Ephemeridenzeit und Weltzeit ($T = (JD-2451545)/36525$)

Zeitraum	$\Delta T = ET - UT$	t
1825 – 1850	$10^s.4 - 80^s.8t + 413^s.9t^2 - 572^s.3t^3$	$T + 1.75$
1850 – 1875	$6^s.6 + 46^s.3t - 358^s.4t^2 + 18^s.8t^3$	$T + 1.50$
1875 – 1900	$-3^s.9 - 10^s.8t - 166^s.2t^2 + 867^s.4t^3$	$T + 1.25$
1900 – 1925	$-2^s.6 + 114^s.1t + 327^s.5t^2 - 1467^s.4t^3$	$T + 1.00$
1925 – 1950	$24^s.2 - 6^s.3t - 8^s.2t^2 + 483^s.4t^3$	$T + 0.75$
1950 – 1975	$29^s.3 + 32^s.5t - 3^s.8t^2 + 550^s.7t^3$	$T + 0.50$
1975 – 2000	$45^s.3 + 130^s.5t - 570^s.5t^2 + 1516^s.7t^3$	$T + 0.25$

eingeben zu müssen, kann man auch auf eine Polynomapproximation der Tabellendaten zurückgreifen.

Für den Zeitraum von 1825 bis 2000 sind dazu in der Tabelle 9.2 verschiedene Polynome aufgeführt, die jeweils Abschnitte von 25 Jahren mit einer typischen Genauigkeit von 1 s abdecken. Da die Genauigkeit außerhalb der angegebenen Zeiträume schnell nachlässt, prüft die Funktion `ETminUT` die jeweiligen Intervallgrenzen und liefert gegebenenfalls in der Variablen `valid` den Wert `false` zurück. Ausgenommen hiervon ist lediglich das letzte Polynom, das auch zur Extrapolation bis ins Jahr 2005 zugelassen wird. Da die $ET-UT$ -Werte aus Beobachtungen bestimmt werden müssen, ist eine längerfristige Vorhersage leider nicht sinnvoll möglich. Die aktuellen Daten werden aber in den verschiedenen Jahrbüchern publiziert und können dort nachgeschlagen werden.

```
-----
// 
// ETminUT: Differenz ET-UT von Ephemeridenzeit und Weltzeit
//   T          Zeit in Julianischen Jahrhunderten seit J2000
//   DTsec      ET-UT in [s]
//   valid      Flag: T im Definitionsbereich der Approximation?
// Beachte: Die zugrundeliegende Näherung deckt die Jahre von 1825 bis 2005 ab
// -----
void ETminUT (double T, double& DTsec, bool& valid)
{
    int i = (int) floor(T/0.25);
    double t = T-i*0.25;
    if ( (T<-1.75) || (0.05<T) ) {
        valid = false;
        DTsec = 0.0;
    }
    else {
        valid = true;
        switch (i) {
            case -7: DTsec=10.4+t*(-80.8+t*( 413.9+t*(-572.3))); break; // 1825-
            case -6: DTsec= 6.6+t*( 46.3+t*(-358.4+t*( 18.8))); break; // 1850-
            case -5: DTsec=-3.9+t*(-10.8+t*(-166.2+t*( 867.4))); break; // 1875-
            case -4: DTsec=-2.6+t*(114.1+t*( 327.5+t*(-1467.4))); break; // 1900-
            case -3: DTsec=24.2+t*(-6.3+t*(-8.2+t*( 483.4))); break; // 1925-
            case -2: DTsec=29.3+t*( 32.5+t*(-3.8+t*( 550.7))); break; // 1950-
        }
    }
}
```

```

case -1: DTsec=45.3+t*(130.5+t*(-570.5+t*( 1516.7))); break; // 1975-
case  0: t+=0.25;
           DTsec=45.3+t*(130.5+t*(-570.5+t*( 1516.7)));           // 2000-
}                                         // 2005
}
}
}

```

Die Differenz von ET und UT ist eine sehr langsam veränderliche Größe und kann deshalb während der gesamten Sonnenfinsternis als konstant angesehen werden.

9.4 Die Dauer der Finsternis

Die Dauer der totalen oder ringförmigen Phase einer Finsternis ist nicht für jeden Punkt der Zentrallinie gleich. Neben dem Durchmesser des Kernschattens hängt sie vor allem von der Geschwindigkeit ab, mit der der Schatten über die Erdoberfläche wandert. Eine exakte Berechnung der Totalitätsdauer erfordert eine iterative Bestimmung der Zeiten, zu denen die totale Phase der Finsternis beginnt und endet und ist deshalb sehr aufwendig. Für die Beurteilung des Finsternisverlaufs und die Auswahl eines günstigen Beobachtungsortes entlang der Zentrallinie ist jedoch auch die folgende Betrachtung völlig ausreichend.

Die äquatorialen Koordinaten

$$\mathbf{r} = (x, y, z) \quad \text{und} \quad \mathbf{r}' = (x', y', z')$$

der Orte, in denen die Schattenachse die Erde zu einer Zeit t und zu einer benachbarten Zeit $t + \Delta t$ schneidet, lassen sich mit den bereits behandelten Gleichungen (9.6) und (9.7) berechnen¹. Die Differenz dieser beiden Vektoren ist ein Maß für die Geschwindigkeit, mit der sich das Kernschattengebiet fortbewegt. Allerdings beschreibt $\mathbf{r}' - \mathbf{r}$ noch nicht die gesuchte Wanderung des Schattens relativ zur Erdoberfläche.

Aus der Dauer von $23^{\text{h}}56^{\text{m}}$ einer vollen Umdrehung ergibt sich der Winkel w , um den sich die Erde in der Zeit Δt um ihre Achse dreht, zu

$$w = 2\pi \cdot \frac{\Delta t}{1436^{\text{m}}} \quad (\text{Bogenmaß}) .$$

Der Ort der Erdoberfläche, der zur Zeit $t + \Delta t$ die äquatorialen Koordinaten $\mathbf{r}' = (x', y', z')$ besitzt, hat zur Zeit t deshalb die Koordinaten

$$\mathbf{r}'' = \begin{pmatrix} +x' \cdot \cos w + y' \cdot \sin w \\ -x' \cdot \sin w + y' \cdot \cos w \\ +z' \end{pmatrix} \approx \begin{pmatrix} x' + wy' \\ y' - wx' \\ z' \end{pmatrix} ,$$

die aus \mathbf{r}' durch eine entsprechende Drehung um die z -Achse mit dem Drehwinkel w hervorgehen. Der Weg, den die Mitte des Kernschattens in der Zeitspanne Δt auf der Erdoberfläche zurücklegt, ist damit

$$\Delta \mathbf{r} = \mathbf{r}'' - \mathbf{r} \tag{9.15}$$

¹Die Abplattung der Erde wird angesichts einiger weiterer Näherungen in diesem Abschnitt vernachlässigt.

oder

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} x' + wy' & - & x \\ y' - wx' & - & y \\ z' & - & z \end{pmatrix} . \quad (9.16)$$

Beschreibt $\mathbf{e} = (e_x, e_y, e_z)$ wie in (9.4) die Richtung der Schattenachse, dann kann $\Delta\mathbf{r}$ in einen Anteil $\Delta\mathbf{r}_{\parallel}$ parallel zu \mathbf{e} und in einen Anteil $\Delta\mathbf{r}_{\perp}$ senkrecht zur Schattenachse zerlegt werden. Die Längen dieser beiden Stücke sind

$$\Delta r_{\parallel} = \Delta\mathbf{r} \cdot \mathbf{e} = \Delta x e_x + \Delta y e_y + \Delta z e_z$$

und

$$\Delta r_{\perp} = \sqrt{(\Delta r)^2 - (\Delta r_{\parallel})^2} .$$

Δr_{\perp} ist die Strecke, um die sich der Auftreffpunkt des Kernschattens in der Zeit Δt senkrecht zur Einfallsrichtung des Schattens weiterbewegt. Die Dauer τ der Totalität oder der ringförmigen Phase beträgt bei einem Durchmesser $|d|$ des Kernschattenkegels demnach

$$\tau = \frac{|d|}{\Delta r_{\perp}} \cdot \Delta t .$$

9.5 Sonnen- und Mondkoordinaten

Eine wichtige Voraussetzung für die Berechnung von Sonnenfinsternissen sind natürlich genaue Koordinaten von Sonne und Mond. Um zum Beispiel die Zentrallinie der Finsternis auf etwa 10 km genau festlegen zu können, muß der Ort des Mondes in seiner Bahn mit einer vergleichbaren Genauigkeit bekannt sein. In einer mittleren Entfernung von 380 000 km entspricht dies einem Winkel von 5''. Ähnlich genau müssen auch die geozentrischen Sonnenkoordinaten bekannt sein. Diese Anforderungen werden von den bereits behandelten Funktionen SunPos und MoonPos sehr gut erfüllt. Auf die Grundlagen der darin verwendeten Störungsreihen soll hier nicht mehr näher eingegangen werden. Sie können in Kap. 6 und 8 noch einmal nachgeschlagen werden.

Für unsere Zwecke ist allerdings neben der hohen Genauigkeit auch eine ausreichende Schnelligkeit wünschenswert, weil man für die Berechnung des Finsternisverlaufs eine große Zahl von Sonnen- und Mondpositionen benötigt. Deshalb soll zunächst eine Tschebyscheff-Entwicklung der einzelnen Koordinaten berechnet werden. Darunter versteht man bestimmte Polynome, die die Sonnen- und Mondbahn über einen kurzen Zeitraum mit genügender Genauigkeit darstellen, aber wesentlich einfacher auszuwerten sind, als die vollständigen Reihenentwicklungen mit unbeschränkter Gültigkeit. Dieser Weg wurde bereits bei der Berechnung der Mondephemeride (Kap. 8) beschritten. Dort sind auch die Grundlagen der Tschebyscheff-Entwicklung und die zugehörige Klasse Cheb3D behandelt, mit denen der Leser bereits vertraut sein sollte.

Die Entwicklung der Mondkoordinaten erfordert lediglich die Deklaration eines Objekts der Klasse `Cheb3D`, wobei im Aufruf des Konstruktors die zu approximierende Funktion, die benötigte Ordnung und das Approximationsintervall anzugeben sind.

```
// Konstanten
const double Interval = 1.0/36525.0; // 1d (in Jahrhunderten)

// Globale Sonnen- und Mondephemeriden
Cheb3D ChebMoonEqu = Cheb3D(MoonEqu,10,Interval);
```

Da sich eine Sonnenfinsternis immer nur über einige Stunden erstreckt, genügt es, bei der Entwicklung mit Polynomen der Ordnung zehn und einem eintägigen Intervall zu arbeiten. Weiterhin bietet es sich an, für die Approximation anstelle von `MoonPos` die Funktion `MoonEqu` zu verwenden, da die Berechnung der Sonnenfinsternis wie beschrieben in einem äquatorialen Koordinatensystem erfolgt. Auf diese Weise sind später keine weiteren Transformationen zwischen dem ekliptikalnen und äquatorialen System mehr erforderlich. Die Polynome für die Mondkoordinaten können mit Hilfe der Methode `ChebMoonEqu.Value(T)` zu einem beliebigen Zeitpunkt ausgewertet werden:

```
r_Moon = ChebMoonEqu.Value(T_ET); // [km]
```

Anzumerken ist noch, daß die eigentliche Berechnung der Tschebyscheff-Approximation selbstständig beim ersten Aufruf von `Value` durchgeführt wird, falls seitens des Benutzers nicht zuvor die Methode `Fit` explizit aufgerufen wurde.

Um auch die Sonnenkoordinaten entsprechend darstellen zu können, benötigen wir noch eine kleine Hilfsfunktion. `SunEqu` berechnet zu einer gegebenen Zeit die wahren äquatorialen Koordinaten der Sonne. Dazu muß neben der Umwandlung der ekliptikalnen Sonnenkoordinaten in äquatoriale Koordinaten auch die Nutation (vgl. Abschn. 6.4.2) berücksichtigt werden. Der resultierende Positionsvektor gibt dann den Ort der Sonne in einem Koordinatensystem wieder, das parallel zur aktuellen (und nicht zur mittleren) Lage der Erdachse und des Erdäquators orientiert ist. Für die Mondbahn wurde die Nutation bereits in entsprechender Weise in der Funktion `MoonEqu` berücksichtigt.

```
-----
// SunEqu: Berechnet die aequatoriale Position der Sonne unter Verwendung einer
//          analytischen Reihenentwicklung
//          T          Zeit in Julianischen Jahrhunderten seit J2000
//          <return> Geozentrische Position der Sonne (in [AE]), bezogen auf
//                      Aequator und Fruehlingspunkt des Datums
//-----
Vec3D SunEqu ( double T )
{
    return NutMatrix(T) * Ecl2EquMatrix(T) * SunPos(T);
}
```

Die Tschebyscheff-Entwicklung der Sonnenkoordinaten erfolgt wieder durch Deklaration eines entsprechenden `Cheb3D` Objekts:

```
Cheb3D ChebSunEqu(SunEqu,5,Interval);
```

Wegen der langsameren Bewegung der Sonne genügt hier allerdings ein niedrigerer Entwicklungsgrad.

Um die Positionen von Sonne und Mond korrekt miteinander vergleichen zu können, ist bei der Berechnung der Sonnenposition mit SunEqu (und somit auch ChebSunEqu.Value) noch die Lichtlaufzeit zu beachten, die von der Sonne zur Erde etwa 8.32 Minuten beträgt. In dieser Zeit wandert die Sonne in ihrer Bahn rund 20" weiter. Da der zu einer bestimmten Zeit t beobachtete Ort der Sonne dem geometrischen Ort zur früheren Zeit $t - 8^{\text{m}}32$ der Lichtaussendung entspricht, muß die Auswertung der Sonnenkoordinaten entsprechend vorverlegt werden.

```
const double tau_Sun = 8.32 / ( 1440.0*36525.0);      // 8.32 min [cy]
r_Sun = ChebSunEqu.Value (T_ET-tau_Sun)*AU;           // [km]
```

Beim Mond ist dieser Effekt bereits in die Brownsche Theorie der Mondbahn mit eingearbeitet und muß deshalb nicht gesondert behandelt werden.

9.6 Das Programm ECLIPSE

Das Programm Eclipse berechnet ausgehend vom Datum des nächstgelegenen Neumonds die Zentrallinie einer Sonnenfinsternis und die Dauer der totalen oder ringförmigen Phase.

Das Kernstück des Programms bildet die Funktion `Intersect`, in der – ausgehend von den geozentrischen Koordinaten von Sonne und Mond – der Schnittpunkt der Schattenachse mit der Erdoberfläche bestimmt wird. Ferner werden die Durchmesser der beiden Schattenkegel in der Nähe der Erde berechnet, aus denen sich auch die jeweilige Phase der Finsternis ergibt. Die Funktion `Central` rechnet die von `Intersect` gelieferten äquatorialen Koordinaten des Kernschattens in geographische Koordinaten um und bestimmt zusätzlich die Dauer der Totalität.

Als Eingaben des Programms werden neben dem Neumonddatum auch die Differenz zwischen Welt- und Ephemeridenzeit und die Ausgabeschriftweite abgefragt. Der Wert $\Delta T = ET - UT$ ist allerdings nur für vergangene Zeitpunkte bekannt und kann ansonsten nur unzuverlässig geschätzt werden. Für Vorhersagen von Finsternissen wird deshalb normalerweise $ET - UT = 0$ gesetzt. Die so berechneten Koordinaten eines Punktes der Zentrallinie unterscheiden sich durch einen Versatz der geographischen Länge von den korrekten Werten. Dieser Unterschied entspricht der Drehung der Erde während der kurzen Zeitspanne ΔT . Ist λ' die berechnete Länge, dann ergibt sich bei der hier gewählten Zählweise ($\lambda > 0$ nach Osten) die tatsächliche geographische Länge zu:

$$\lambda = \lambda' + 0^{\circ}25/\text{m} \cdot \Delta T .$$

Bei dem gegenwärtigen Wert von $\Delta T \approx 1^{\text{m}}$ liegen die mit $\Delta T = 0$ vorhergesagten Koordinaten der Zentrallinie also um $1/4^{\circ}$ zu weit westlich. Am Äquator entspricht dies einer Strecke von rund 28 km.

```

//-----
// Datei: Eclipse.cpp
// Zweck: Berechnung der Zentrallinie und Dauer von Sonnenfinsternissen
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//-----

#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>

#include "APC_Cheb.h"
#include "APC_Const.h"
#include "APC_IO.h"
#include "APC_Math.h"
#include "APC_Moon.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"

using namespace std;

// Typen
enum enPhase { NoEclipse, partial, NonCenAnn, NonCenTot, annular, total };

// Konstanten
const double Interval = 1.0/36525.0; // 1d in Jahrhunderten
// Globale Daten fuer Sonnen- und Mondpositionen
Cheb3D ChebMoonEqu(MoonEqu,10,Interval);
Cheb3D ChebSunEqu (SunEqu, 5,Interval);

//-----
// Intersect: Berechnet den Schnittpunkt der Schattenachse mit der
// Erde, den Einheitsvektor in Richtung der Schattenachse sowie
// den Schattendurchmesser und die Phase der Finsternis
// T_ET      Zeit (ET) in Julianischen Jahrhunderten seit J2000
// r         Schnittpunkt der Schattenachse mit der Erdoberflaeche;
//           geozentrische Koordinaten in [km]
// e         Einheitsvektor in Richtung der Schattenachse
// D_umbra   Schattendurchmesser auf der Fundamentalebene in [km]
// Phase     Phase der Finsternis
//-----

void Intersect ( double T_ET,
                 Vec3D& r, Vec3D& e, double& D_umbra, enPhase& Phase )
{
    // Konstanten
    const double tau_Sun = 8.32 / ( 1440.0*36525.0); // 8.32 min in Jahrhunderten
    const double fac      = 0.996647; // Verhaeltnis polarer/aequat. Erdradius
    // Variablen
    double r_MS,s0,Delta,r0,s;
    double D_penumbra;
    Vec3D r_Moon;
    Vec3D r_Sun;
}

```

```

// Positionen von Sonne und Mond bezogen auf Aequator und Aequinoktium des
// Datums (Sonnenposition beruecksichtigt die Lichtlaufzeit)
r_Moon = ChebMoonEqu.Value(T_ET); // [km]
r_Sun = ChebSunEqu.Value (T_ET-tau_Sun)*AU; // [km]
// z-Komponente strecken, um die Erdabplattung zu beruecksichtigen
r_Moon = Vec3D( r_Moon[x], r_Moon[y], r_Moon[z]/fac );
r_Sun = Vec3D( r_Sun[x] , r_Sun[y] , r_Sun[z]/fac );
// Achse des Schattens (Einheitsvektor Sonne->Mond)
r_MS = Norm(r_Moon-r_Sun);
e = (r_Moon-r_Sun)/r_MS;
// Entfernung des Mondes von der Fundamentalebene
s0 = -Dot(r_Moon,e);
// Entfernung der Schattenachse vom Erdmittelpunkt
Delta = s0*s0 + R_Earth*R_Earth - Dot(r_Moon,r_Moon);
r0 = sqrt(R_Earth*R_Earth-Delta);
// Durchmesser von Kern- und Halbschatten auf der Fundamentalebene
D_umbra = 2.0*((R_Sun-R_Moon)*(s0/r_MS)-R_Moon);
D_penumbra = 2.0*((R_Sun-R_Moon)*(s0/r_MS)+R_Moon);

// Phase bestimmen und eventuell Schattenkoordinaten berechnen
if (r0<R_Earth) {
    // Schattenachse schneidet die Erde; ->totale oder ringfoermige Finsternis:
    // Schnittpunkt von Schattenachse und Erdoberflaeche
    s = s0-sqrt(Delta);
    r = r_Moon + s*e;
    // z-Komponente stauchen
    r = Vec3D(r[x],r[y],fac*r[z]);
    // Kernschattendurchmesser an der Erdoberflaeche
    D_umbra = 2.0*((R_Sun-R_Moon)*(s/r_MS)-R_Moon);
    Phase = ( (D_umbra>0.0)? annular : total );
}
else {
    if ( r0 < R_Earth+0.5*fabs(D_umbra) ) {
        // nichtzentrale Finsternis
        Phase = ( (D_umbra>0.0)? NonCenAnn : NonCenTot );
    }
    else {
        // Partielle oder keine Finsternis
        Phase = ( (r0<R_Earth+0.5*fabs(D_penumbra))? partial : NoEclipse );
    };
    r = Vec3D(); // Platzhalter (Nullvektor)
};

//-----
// Central: Zentrallinie, Dauer und Phase der Finsternis
// MjdUT Weltzeit (UT) als MJD
// ET_UT Differenz zwischen Ephemeridenzeit (ET) und Weltzeit in [s]
// Lambda Geographische Laenge (oestl. pos.) der Schattenmitte in [rad]
// Phi Geographische Breite der Schattenmitte in [rad]
// Durat Dauer der totalen/ringfoermigen Phase an diesem Ort in [min]
// Phase Phase der Finsternis
//-----
void Central ( double MjdUT, double ET_UT,
               double &Lambda, double& Phi, double& Durat, enPhase& Phase )

```

```

{
    // Konstanten
    const double dt      = 0.1 / (1440.0*36525.0); // 0.1 min in Jahrhunderten
    const double omega = pi2 * 1.002738*36525.0; // [rad/Jahrhundert]
    // Variablen
    double T_ET;           // Ephemeridenzeit
    Vec3D r,r_G,rr,dr;     // Koordinaten des Schattens
    Vec3D e,ee;            // Schattenachse
    double D_umbra, DU;    // Schattendurchmesser
    enPhase Ph=NoEclipse;  // Phase
    double w;              // Drehwinkel der Erde
    double drp;             // differentielle Verschiebung des Schattens

    // Ephemeridenzeit
    T_ET = ( MjdUT + ET_UT/86400.0 - MJD_J2000 ) / 36525.0;
    // Schnittpunkt der Schattenachse mit der Erdoberflaeche
    Intersect ( T_ET, r, e, D_umbra, Phase );
    // Nur fuer zentrale Phase: geogr. Koordinaten und Dauer der zentralen Phase
    Lambda = Phi = Durat = 0.0;
    if ( Phase >= annular ) {
        // Geographische Koordinaten des Schattenmittelpunkts
        r_G     = R_z(GMST(MjdUT))*r;           // Greenwich Koordinaten
        Lambda = Modulo(r_G[phi]+pi,2*pi)-pi;    // oestl. Laenge
        Phi    = r_G[theta];                     // Geozentrische Breite
        Phi    = Phi + 0.1924*Rad*sin(2*Phi);    // Geographische Breite
        // Dauer der zentralen Phase an diesem Ort
        // (a) Koordinaten des Schattens zur Zeit T+dt (oder T-dt)
        if (Ph<annular) {
            Intersect ( T_ET-dt, rr, ee, DU, Ph ); w = -dt*omega;
        }
        else {
            Intersect ( T_ET+dt, rr, ee, DU, Ph ); w = +dt*omega;
        }
        // (b) Verschiebung dr des Schattens auf der Erdoberflaeche und
        //     Anteil drp senkrecht zur Schattenachse
        dr = Vec3D ( rr[x]-r[x]+w*r[y], rr[y]-r[y]-w*r[x], rr[z]-r[z] );
        drp = Norm ( dr - Dot(dr,e)*e );
        Durat = (36525.0*1440.0)*dt * fabs(D_umbra) / drp; // [min]
    };
};

//-----
// GetInput: Abfrage des ungefaehren Neumondzeitpunkts, der Ausgabeschrittweite
// und von ET-UT
// MjdStart Startzeitpunkt zur Berechnung der Finsternis
// Step    Ausgabeschrittweite in [d]
// MjdEnd  Endzeitpunkt zur Berechnung der Finsternis
// ET_UT   Differenz Ephemeridenzeit - Weltzeit (ET-UT) in [s]
//-----
void GetInput(double& MjdStart, double& Step, double& MjdEnd, double& ET_UT)
{
    // Variablen
    int year, month, day, step;
    double hour, MJD;
    bool valid;
}

```

```

// Neumonddatum und Schrittweite
cout << endl
    << " Neumonddatum (JJJJ MM TT HH.H)           ... ";
cin >> year >> month >> day >> hour; cin.ignore(81,'\'n');

cout << " Ausgabeschrittweite (min)" << setw(18) << right << "... ";
cin >> step; cin.ignore(81,'\'n');
// Neumonddatum auf Ausgabeschrittweite runden
Step = step/1440.0; // [d]
MJD = Mjd(year,month,day) + floor((hour/24.0)/Step+0.5)*Step;
// Suchzeitraum (Neumond +/- 6 Stunden)
MjdStart = MJD - 0.25;
MjdEnd   = MJD + 0.25;
// Differenz Ephemeridenzeit - Weltzeit
ETminUT ( (MJD-MJD_J2000)/36525.0, ET_UT, valid );
if ( valid ) {
    cout << " Differenz ET-UT (Vorschlag:" << fixed << setw(6)
        << setprecision (1) << ET_UT << " sec) ... ";
    cin >> ET_UT; cin.ignore(81,'\'n');
}
else {
    cout << " Differenz ET-UT (sec)" << setw(21) << right << "... ";
    cin >> ET_UT; cin.ignore(81,'\'n');
}

//-----
// 
// Hauptprogramm
// 
//-----
void main(int argc, char* argv[]) {

    // Variablen
    double    MjdStart, Step, MjdEnd;
    double    MjdUT, ET_UT;
    double    Lambda, Phi, Durat;
    enPhase   Phase;
    char      OutputFile[APC_MaxFilename] = "";
    ofstream  OutFile;

    // Titel
    cout << endl
        << "     ECLIPSE: Zentrallinie und Dauer von Sonnenfinsternissen" << endl
        << "             (c) 1999 Oliver Montenbruck, Thomas Pfleger   " << endl
        << endl;
    // Abfrage der Eingaben
    GetInput ( MjdStart, Step, MjdEnd, ET_UT );
    // Eventuell in Ausgabedatei umlenken
    if (argc==2) {
        strcpy(OutputFile, argv[1], APC_MaxFilename);
        OutFile.open(OutputFile);
        if (!OutFile.is_open())
            cout = OutFile;
    }
}

```

```

// Vorspann
cout << endl << endl
    << "    Datum      UT          Phi       Lambda   Dauer   Phase " << endl
    << "          h   m           o   '       o   '     min           " << endl
    << endl;

// Phase und Zentrallinie der Finsternis
MjdUT = MjdStart;
// Schleife ueber Berechnungszeitpunkte
while ( MjdUT < MjdEnd + Step/2 ) {
    // Phase und Ort des Schattens
    Central ( MjdUT, ET_UT, Lambda, Phi, Durat, Phase );
    // Ausgabe
    if ( Phase != NoEclipse ) {
        cout << " " << DateTime(MjdUT,HHMM);
        if ( Phase < annular )
            cout << "    -- --    -- --    ---";
        else {
            cout << showpos
                << "      " << setw(6) << Angle(Deg*Phi,DMM)
                << "      " << setw(7) << Angle(Deg*Lambda,DMM)
                << noshowpos
                << fixed << setprecision(1) << setw(7) << Durat;
        };
        switch(Phase) {
            case partial : cout << "    partiell           "; break;
            case NonCenAnn: cout << "    ringfoermig (nicht zentral)"; break;
            case NonCenTot: cout << "    total (nicht zentral)      "; break;
            case annular : cout << "    ringfoermig           "; break;
            case total    : cout << "    total                 ";
        };
        cout << endl;
    };
    MjdUT += Step; // Naechster Berechnungszeitpunkt
};
if (OutFile.is_open()) OutFile.close();
}

```

Als Beispiel wollen wir den Verlauf einer totalen Sonnenfinsternis berechnen lassen, die 1999 von Süddeutschland aus zu sehen sein wird. Die Zentrallinie dieser Finsternis verläuft in unmittelbarer Nähe der Städte Stuttgart und München. Zunächst rufen wir dazu Phases auf, um die Neumondzeiten des Jahres 1999 zu bestimmen und zu prüfen, an welchen Daten eine Finsternis möglich ist:

```

PHASES: Zeitpunkte von Mondphasen und Finsternissen
(c) 1999 Oliver Montenbruck, Thomas Pfleger

```

```
Bestimmung der Mondphasen fuer das Jahr ... 1999
```

Neumond	Erstes Viertel	Vollmond	Letztes Viertel
1998/12/18 22:42	1998/12/26 10:46	1999/01/02 02:50	1999/01/09 14:21
1999/01/17 15:46	1999/01/24 19:15	1999/01/31 16:06p?	1999/02/08 11:58
<u>1999/02/16 06:39c</u>	1999/02/23 02:43	1999/03/02 06:58	1999/03/10 08:40

1999/03/17 18:48	1999/03/24 10:18	1999/03/31 22:49	1999/04/09 02:51
1999/04/16 04:22	1999/04/22 19:02	1999/04/30 14:55	1999/05/08 17:28
1999/05/15 12:05	1999/05/22 05:34	1999/05/30 06:40	1999/06/07 04:20
1999/06/13 19:03	1999/06/20 18:13	1999/06/28 21:37	1999/07/06 11:57
1999/07/13 02:24	1999/07/20 09:00	<u>1999/07/28 11:25p</u>	1999/08/04 17:27
<u>1999/08/11 11:08c</u>	1999/08/19 01:47	<u>1999/08/26 23:48</u>	1999/09/02 22:17
1999/09/09 22:02	1999/09/17 20:06	1999/09/25 10:51	1999/10/02 04:02
1999/10/09 11:34	1999/10/17 15:00	1999/10/24 21:02	1999/10/31 12:04
1999/11/08 03:53	1999/11/16 09:03	1999/11/23 07:04	1999/11/29 23:18
1999/12/07 22:32	1999/12/16 00:50	1999/12/22 17:31	1999/12/29 14:04
2000/01/06 18:14	2000/01/14 13:34	<u>2000/01/21 04:40t</u>	2000/01/28 07:57

Alle Zeitangaben in Weltzeit (UT).

Die Kennzeichnung „c“ am 16. Februar und am 11. August dieses Jahres zeigt an, daß die ekliptikale Breite des Mondes an diesen Neumondtagen klein genug ist, um das Auftreten zentraler Finsternisse sicher vorhersagen zu können (vgl. Tabelle 9.1). Wie eine genauere Untersuchung mit **Eclipse** zeigt, ergeben sich folgende Sichtbarkeitsgebiete:

16.02.1999: Südlicher Indischer Ozean, Prince Edward Islands, Iles Crozet, Australien (ringförmig);

11.08.1999: Nordatlantik, Südliches Großbritannien, Mitteleuropa, Náher Osten, Indien (total).

Der Verlauf der totalen Finsternis vom 11. August 1999 soll nun mit **Eclipse** bestimmt werden. Dazu geben wir das bereits bestimmte Neumonddatum, die gewünschte Schrittweite in Minuten sowie einen Wert für die Differenz „Ephemeridenzeit – Weltzeit“ ein. Für die gesuchte Finsternis übernehmen wir den vorgeschlagenen Schätzwert von 65°.5. Diese und die übrigen Eingaben sind im folgenden kursiv dargestellt. Im folgenden Ausdruck sind einige Werte zur Zeit der partiellen Phase nicht aufgeführt, um die Ausgabe nicht unnötig zu verlängern.

ECLIPSE: Zentrallinie und Dauer von Sonnenfinsternissen
 (c) 1999 Oliver Montenbruck, Thomas Pfleger

Neumonddatum (JJJJ MM TT HH.H) ... 1999 08 11 11.1
 Ausgabeschrittweite(min) ... 3.0
 Differenz ET-UT (Vorschlag: 65.5 sec) ... 65.5

Datum	UT	Phi	Lambda	Dauer	Phase
	h m	o ,	o ,	min	
1999/08/11 08:27	-- --	-- --	-- --	---	partiell
1999/08/11 08:30	-- --	-- --	-- --	---	partiell
.
1999/08/11 09:24	-- --	-- --	-- --	---	partiell
1999/08/11 09:27	-- --	-- --	-- --	---	partiell
1999/08/11 09:30	-- --	-- --	-- --	---	total (nicht zentral)
1999/08/11 09:33	+44 55	- 50 20		1.1	total

1999/08/11 09:36	+46 26	- 43 36	1.2	total
1999/08/11 09:39	+47 26	- 38 27	1.4	total
1999/08/11 09:42	+48 11	- 34 06	1.5	total
1999/08/11 09:45	+48 45	- 30 15	1.5	total
1999/08/11 09:48	+49 12	- 26 45	1.6	total
1999/08/11 09:51	+49 33	- 23 31	1.7	total
1999/08/11 09:54	+49 48	- 20 29	1.8	total
1999/08/11 09:57	+50 00	- 17 38	1.8	total
1999/08/11 10:00	+50 08	- 14 56	1.9	total
1999/08/11 10:03	+50 12	- 12 22	1.9	total
1999/08/11 10:06	+50 14	- 9 54	2.0	total
1999/08/11 10:09	+50 13	- 7 33	2.1	total
1999/08/11 10:12	+50 09	- 5 17	2.1	total
1999/08/11 10:15	+50 04	- 3 05	2.1	total
1999/08/11 10:18	+49 56	- 0 59	2.2	total
1999/08/11 10:21	+49 46	+ 1 03	2.2	total
1999/08/11 10:24	+49 35	+ 3 02	2.3	total
1999/08/11 10:27	+49 22	+ 4 56	2.3	total
1999/08/11 10:30	+49 08	+ 6 48	2.3	total
1999/08/11 10:33	+48 52	+ 8 36	2.3	total
1999/08/11 10:36	+48 34	+ 10 21	2.4	total
1999/08/11 10:39	+48 16	+ 12 03	2.4	total
1999/08/11 10:42	+47 56	+ 13 42	2.4	total
1999/08/11 10:45	+47 35	+ 15 20	2.4	total
1999/08/11 10:48	+47 12	+ 16 54	2.4	total
1999/08/11 10:51	+46 49	+ 18 27	2.4	total
1999/08/11 10:54	+46 24	+ 19 57	2.4	total
1999/08/11 10:57	+45 59	+ 21 26	2.4	total
1999/08/11 11:00	+45 33	+ 22 53	2.4	total
1999/08/11 11:03	+45 05	+ 24 18	2.4	total
1999/08/11 11:06	+44 37	+ 25 42	2.4	total
1999/08/11 11:09	+44 08	+ 27 04	2.4	total
1999/08/11 11:12	+43 38	+ 28 26	2.4	total
1999/08/11 11:15	+43 06	+ 29 46	2.4	total
1999/08/11 11:18	+42 35	+ 31 06	2.4	total
1999/08/11 11:21	+42 02	+ 32 24	2.4	total
1999/08/11 11:24	+41 28	+ 33 42	2.4	total
1999/08/11 11:27	+40 53	+ 35 00	2.3	total
1999/08/11 11:30	+40 18	+ 36 17	2.3	total
1999/08/11 11:33	+39 42	+ 37 34	2.3	total
1999/08/11 11:36	+39 04	+ 38 52	2.3	total
1999/08/11 11:39	+38 26	+ 40 09	2.2	total
1999/08/11 11:42	+37 47	+ 41 28	2.2	total
1999/08/11 11:45	+37 06	+ 42 46	2.1	total
1999/08/11 11:48	+36 25	+ 44 06	2.1	total
1999/08/11 11:51	+35 42	+ 45 27	2.1	total
1999/08/11 11:54	+34 59	+ 46 50	2.0	total
1999/08/11 11:57	+34 14	+ 48 15	2.0	total
1999/08/11 12:00	+33 27	+ 49 42	1.9	total
1999/08/11 12:03	+32 39	+ 51 12	1.9	total
1999/08/11 12:06	+31 49	+ 52 46	1.8	total
1999/08/11 12:09	+30 57	+ 54 25	1.7	total
1999/08/11 12:12	+30 03	+ 56 08	1.7	total
1999/08/11 12:15	+29 06	+ 57 59	1.6	total
1999/08/11 12:18	+28 06	+ 59 59	1.5	total

1999/08/11 12:21	+27 02	+ 62 10	1.5	total
1999/08/11 12:24	+25 53	+ 64 37	1.4	total
1999/08/11 12:27	+24 37	+ 67 27	1.3	total
1999/08/11 12:30	+23 10	+ 70 53	1.2	total
1999/08/11 12:33	+21 21	+ 75 33	1.0	total
1999/08/11 12:36	-- --	-- --	--	total (nicht zentral)
1999/08/11 12:39	-- --	-- --	--	partiell
1999/08/11 12:42	-- --	-- --	--	partiell
.
1999/08/11 13:36	-- --	-- --	--	partiell
1999/08/11 13:39	-- --	-- --	--	partiell

Wenn die Phase der Finsternis total, aber nicht zentral ist, fällt die Achse des Kernschattenkegels an der Erde vorbei. Es existiert aber dennoch ein Gebiet auf der Erde, das vom Kernschattenkegel geschnitten wird. Dies ist meist zu Beginn und am Ende einer Finsternis der Fall. Sehr selten finden auch Finsternisse statt, die zwar total oder ringförmig, aber nirgends zentral erscheinen, da der Schattenkegel des Mondes die Erde nur tangential streift. Finsternisse dieser Art lohnen die Beobachtung wegen ihrer kurzen Dauer nicht und finden zudem meist in hohen geographischen Breiten statt.

9.7 Die lokalen Umstände einer Sonnenfinsternis

Neben der Lage der Zentrallinie sind für die Beobachtung einer Sonnenfinsternis Angaben über die lokalen Umstände von Interesse, die den Finsternisverlauf für einen gegebenen Ort der Erde näher beschreiben.

Hierzu zählen neben der maximalen Phase, Größe und Bedeckung insbesondere die Kontaktzeiten, also die Zeitpunkte, zu denen sich die Ränder von Sonne und Mond berühren. Als ersten und vierten Kontakt bezeichnet man dabei den Ein- und Austritt in den Halbschattenkegel und damit Beginn und Ende der partiellen Phase beziehungsweise der gesamten Finsternis. Entsprechend beschreiben der zweite und dritte Kontakt (die auch als innere Kontakte bezeichnet werden) den Zeitraum der totalen oder ringförmigen Phase, falls der Beobachter sich an einem Ort befindet, der vom Kernschattenkegel des Mondes überstrichen wird.

Üblicherweise werden die Kontaktzeiten noch um die Angabe der Positionswinkel ergänzt. Der von Norden über Osten positiv gezählte Positionswinkel gibt beim ersten und vierten Kontakt an, an welcher Stelle des Sonnenrandes sich der Mond erstmals vor die Sonnenscheibe schiebt bzw. wo er sie schließlich wieder freigibt. Die Positionswinkel der beiden inneren Kontakte bezeichnen die Stellen des Verschwindens und erneuten Sichtbarwerdens der Sonnenscheibe am Mondrand (bei einer totalen Finsternis) oder die Orte, an denen sich bei einer ringförmigen Finsternis der Ring erstmals schließt bzw. wieder öffnet.

Die geometrische Bedingung für den ersten oder vierten Kontakt einer Sonnenfinsternis ist, daß sich der Beobachter irgendwo auf der Mantelfläche des Halbschattenkegels des Mondes befindet. Der zweite oder dritte Kontakt findet dagegen statt, wenn sich der Beobachter auf der Mantelfläche des Kernschattenkegels befindet.

Um diesen Sachverhalt zu beschreiben, verwendet man wie bei der Behandlung von Sternbedeckungen (vgl. Abschn. 10.3) das Koordinatensystem der Fundamentalebene, die senkrecht zur Schattenachse durch den Erdmittelpunkt verläuft (Abb. 9.4). Die x -Achse und die y -Achse spannen dabei die Fundamentalebene auf, während die z -Achse in Richtung der Verbindungsgeraden von Sonne und Mond weist. Die x -Achse ist ferner so gewählt, daß sie entlang der Schnittlinie von Erdäquator und Fundamentalebene verläuft. Damit gelten für die Einheitsvektoren \mathbf{i} , \mathbf{j} und \mathbf{k} in Richtung der x -, y - und z -Achse des Koordinatensystems der Fundamentalebene die Beziehungen

$$\mathbf{i} = \begin{pmatrix} -k_y / \sqrt{k_x^2 + k_y^2} \\ +k_x / \sqrt{k_x^2 + k_y^2} \\ 0 \end{pmatrix} \quad \mathbf{j} = \mathbf{k} \times \mathbf{i} \quad \mathbf{k} = \frac{\mathbf{r}_\odot - \mathbf{r}_M}{|\mathbf{r}_\odot - \mathbf{r}_M|} , \quad (9.17)$$

wobei die Vektoren \mathbf{r}_\odot und \mathbf{r}_M die äquatorialen Koordinaten von Sonne und Mond bezeichnen. Durch Projektion von \mathbf{r}_M auf die Basisvektoren \mathbf{i} , \mathbf{j} und \mathbf{k} erhält man damit die Koordinaten

$$\begin{pmatrix} x_M \\ y_M \\ z_M \end{pmatrix} = \begin{pmatrix} \mathbf{r}_M \cdot \mathbf{i} \\ \mathbf{r}_M \cdot \mathbf{j} \\ \mathbf{r}_M \cdot \mathbf{k} \end{pmatrix}$$

des Mondes im System der Fundamentalebene.

Da die Verbindungsgerade von Sonne und Mond definitionsgemäß parallel zur z -Achse verläuft, bezeichnen x_M und y_M gleichzeitig die x - und y -Koordinate der Schattenachse in der Fundamentalebene. Aus der Orientierung der Fundamentalebene folgt darüberhinaus, daß die beiden Schattenkegel von Umbra und Penumbra die Fundamentalebene in einem Kreis schneiden, dessen Radius nach (9.2) durch

$$l = z_M \frac{R_\odot \mp R_M}{r_{\odot M}} \mp R_M \quad (9.18)$$

gegeben ist. Hierin sind R_\odot und R_M die Radien von Sonne und Mond, während $r_{\odot M} = |\mathbf{r}_\odot - \mathbf{r}_M|$ deren gegenseitige Entfernung bezeichnet. Das negative Vorzeichen bezieht sich dabei auf den Umbrakegel, während das positive Vorzeichen für den Penumbrikegel gilt.

Um festzustellen, wann sich der Beobachter auf der Mantelfläche eines der beiden Schattenkegel befindet, wird zusätzlich die geozentrische Beobachterposition auf die Fundamentalebene abgebildet. Nach Abschn. 9.3 hat ein Beobachter mit der geozentrischen Länge λ , der geozentrischen Breite φ' und der Entfernung r vom Erdmittelpunkt die äquatorialen Koordinaten

$$\mathbf{r}_B = \mathbf{R}_z(-\Theta_0(t)) \begin{pmatrix} r \cos(\varphi') \cos(\lambda) \\ r \cos(\varphi') \sin(\lambda) \\ r \sin(\varphi') \end{pmatrix} , \quad (9.19)$$

wobei $\Theta_0(t)$ die Greenwich-Sternzeit bedeutet. Die Koordinaten (x_B, y_B, z_B) des Beobachters im System der Fundamentalebene erhält man entsprechend wieder

durch Projektion von \mathbf{r}_B auf die Basisvektoren \mathbf{i} , \mathbf{j} und \mathbf{k} . Aus der Höhe z_B des Beobachters über der Fundamentalebene läßt sich nun mit zusätzlicher Kenntnis des halben Öffnungswinkels f des Schattenkegels und seines Durchmessers l auf der Fundamentalebene der Radius

$$L = l - z_B \tan f \quad (9.20)$$

des Schattenkegels am Ort des Beobachters berechnen. Für partielle und ringförmige Finsternisse ist L immer positiv, wohingegen der Radius des Umbragebietes im Falle einer totalen Finsternis konventionsgemäß negativ wird.

Als Bedingung für die Lage des Beobachters auf dem Schattenkegel läßt sich damit formulieren, daß der Abstand des Beobachters von der Schattenachse betragsmäßig gleich dem Radius des Schattenkegels am Ort des Beobachters ist. Äquivalent hierzu ist die Bedingung, daß die Abstandsfunktion

$$f(t) = (x_M - x_B)^2 + (y_M - y_B)^2 - L^2 \quad (9.21)$$

den Wert Null annimmt. Die Kontaktzeitpunkte lassen sich demnach iterativ durch Bestimmung der Nullstellen von $f(t)$ ermitteln. Der Zeitpunkt der maximalen Finsternis ergibt sich entsprechend aus der Lage des Minimums von $f(t)$.

Da die relative Lage von Beobachter und Schattenachse in der Fundamentalebene ein direktes Abbild der scheinbaren Position von Sonne und Mond am Himmel des Beobachters ist, lassen sich die Positionswinkel der Kontaktstellen auf einfache Weise aus den bekannten Fundamentalebenenkoordinaten bestimmen. Zählt man den Positionswinkel des Berührungsproduktes bezüglich der Sonnenmitte von Norden ausgehend positiv in Richtung Osten, dann gilt unabhängig vom Finsternistyp für innere und äußere Kontakte die Beziehung

$$\begin{pmatrix} \cos P \\ \sin P \end{pmatrix} = \begin{pmatrix} (y_M - y_B)/L \\ (x_M - x_B)/L \end{pmatrix} \quad . \quad (9.22)$$

Bei der Auflösung dieser Gleichungen nach P ist zu beachten, daß der Radius L im Falle einer totalen Finsternis wie erwähnt negative Werte annimmt.

Neben dem auf die Nordrichtung bezogenen Positionswinkel P ist bei Sonnenfinsternissen auch die Angabe des auf die Zenitrichtung des Beobachters bezogenen Positionswinkels $V = P - C$ gebräuchlich. Der parallaktische Winkel C entspricht dabei dem Positionswinkel des Zenits und kann mit Hilfe der Beziehung

$$\begin{pmatrix} \cos C \\ \sin C \end{pmatrix} \approx \begin{pmatrix} y_B / \sqrt{x_B^2 + y_B^2} \\ x_B / \sqrt{x_B^2 + y_B^2} \end{pmatrix} \quad . \quad (9.23)$$

ausreichend genau aus den Beobachterkoordinaten auf der Fundamentalebene bestimmt werden.

Zur näheren Charakterisierung des maximalen Verfinsterungsgrades einer Sonnenfinsternis wird bei der Angabe der lokalen Umstände die sogenannte Größe der Finsternis M verwendet. Sie ist definiert als der Bruchteil des scheinbaren Sonnendurchmessers, der zum Zeitpunkt der größten Verfinsterung vom Mond bedeckt

wird. M kann direkt aus den Radien L_1 und L_2 des Penumbrakegels und des Umbrakegels am Ort des Beobachters berechnet werden. Für totale oder ringförmige Finsternisse entspricht sie dem Verhältnis

$$M_2 = (L_1 - L_2)/(L_1 + L_2)$$

der scheinbaren Radien von Sonne und Mond, während die Größe

$$M_1 = (L_1 - m)/(L_1 + L_2)$$

einer partiellen Finsternis zusätzlich vom Abstand m des Beobachters von der Schattenachse zum Maximumszeitpunkt abhängt. Zu beachten ist, daß die Größe M nicht gleichbedeutend mit dem bedeckten Flächenbruchteil der Sonnenscheibe ist und insbesondere bei totalen Finsternissen Werte größer als eins annimmt.

9.8 Das Programm ECLTIMER

Während der Zentraillinienverlauf einer Sonnenfinsternis oft schon Jahre im voraus bekannt und veröffentlicht ist, findet man in Jahrbüchern üblicherweise keine Angaben über die lokalen Umstände einer Finsternis für einen bestimmten Beobachtungsort. Wir wollen deshalb hier in Ergänzung zu *Eclipse* das Programm *EclTimer* vorstellen, mit dem sich die lokalen Umstände einer Sonnenfinsternis für beliebige Orte berechnen lassen.

Aus Platzgründen soll hier nur der prinzipielle Aufbau des Programms beschrieben werden, der sich an dem im vorangehenden Abschnitt dargestellten Rechenweg orientiert. Der vollständige, ausführlich kommentierte Quelltext ist auf der CD zum Buch enthalten.

Zur Bestimmung der Kontaktzeiten werden in der Funktion *Contacts* die Nullstellen der Schattenabstandsfunktion $f(t) = m^2(t) - L^2(t)$ bestimmt, wobei m den Abstand des Beobachters von der Schattenachse und L den Radius des Schattenkegels bezeichnet. Zur Berechnung des Schattenabstands dient die Funktion *ShadowDist*, die ihrerseits auf die beiden Funktionen *Bessel* und *Observer* zurückgreift, in denen die Orientierung der Fundamentalebene, die Parameter des Schattenkegels und die Beobachterkoordinaten ermittelt werden.

Mit Hilfe der quadratischen Interpolation wird zunächst nach dem 1. und 4. Kontakt gesucht, bei dem der Beobachter auf dem Halbschattenkegel liegt. Hierzu wird die in Kap. 3 behandelte Interpolationsfunktion *Quad* verwendet, die es auch ermöglicht den Zeitpunkt zu bestimmen, an dem der Abstand zwischen Beobachter und Schattenachse minimal wird. Ist die Finsternis wenigstens partiell, so berechnet *Contacts* für den Zeitpunkt des Finsternismaximums die Größe der Finsternis und die Bedeckung der Sonne durch den Mond.

Ergibt sich aus dem Abstand des Beobachters von der Schattenachse und dem Kernschattendurchmesser zum Zeitpunkt des Maximums, daß die Finsternis total oder ringförmig ist, so können anschließend die Zeiten des 2. und 3. Kontakts bestimmt werden. Da die Dauer der totalen oder ringförmigen Phase nie länger als ca. 13 Minuten dauert, liegen der 2. und 3. Kontakt jeweils in einem entsprechenden Intervall vor bzw. nach dem Maximum. Mit diesem Wissen kann man

zur Suche nach den Kontaktzeiten ein Verfahren wie die *regula falsi* anwenden, das schneller als die quadratische Interpolation zum Ziel führt. In **EclTimer** wird hierzu eine als Pegasus-Verfahren bezeichnete Variante der *regula falsi* verwendet (vgl. Abschn. 10.2).

Abschließend werden mit Hilfe der Funktion **PosAngles** die Positionswinkel der inneren und äußeren Kontakte in Bezug auf die Nordrichtung und die Zenitrichtung berechnet.

Nach dem Programmstart erfragt **EclTimer** zunächst das Neumonddatum und die Differenz $\Delta T = ET - UT$. Danach sind die geographischen Koordinaten des Beobachtungsortes einzugeben. Eine Übersicht über mögliche Finsterniszeitpunkte kann man bei Bedarf mit dem Programm **Phases** erhalten.

Als Beispiel wollen wir die lokalen Umstände der ringförmigen Sonnenfinsternis vom 10. Mai 1994 berechnen. Der Beobachtungsort sei Rabat in Marokko mit den geographischen Koordinaten $\lambda = 6.8333^\circ$ West und $\varphi = 33.95^\circ$. Als Schätzwert für die Zeitdifferenz ΔT wählen wir den vorgeschlagenen Wert von 60^s . Alle Eingaben sind im folgenden Ausdruck kursiv dargestellt.

ECLTIMER: Lokale Umstände einer Sonnenfinsternis
(c) 1999 Oliver Montenbruck, Thomas Pfleger

Neumonddatum (JJJJ MM TT HH.HHH)	... 1994 05 10 17.1
Differenz ET-UT (Vorschlag: 60.2 sec)	... 60.0
Stationskoordinaten: Laenge (oestl. pos.) [Grad]	... -6.8333
Breite [Grad]	... 33.95

Ringförmige Finsternis mit M=0.931 (0.87).

Maximum: 1994/05/10 18:58:42 UT.

	h m s [UT]	P [o]	V [o]
1. Kontakt:	1994/05/10 17:50:47	267	207
2. Kontakt:	1994/05/10 18:56:40	235	179
3. Kontakt:	1994/05/10 19:00:29	120	65
4. Kontakt:	1994/05/10 19:59:19	89	38

Während der ringförmigen Phase, die für den gewählten Beobachtungsort rund vier Minuten dauert, ist der scheinbare Monddurchmesser um 7% kleiner als der scheinbare Durchmesser der Sonne. Insgesamt sind damit nur rund 87% der scheinbaren Sonnenfläche vom Mond bedeckt. Da die Finsternis am Abend des Tages stattfindet, sind die auf die Zenitrichtung bezogenen Positionswinkel (V) um rund 50° – 60° kleiner als die auf die Nordrichtung bezogenen Werte (P). Anzumerken ist noch, daß die Sonne am Tag der Finsternis in Rabat um $19^\text{h}18^\text{m}$ UT untergeht, so daß das Finsternisende selbst nicht mehr beobachtet werden kann.

10. Sternbedeckungen

Während eines Tages durchwandert der Mond ein rund 13° langes Stück seiner Bahn, die ihn von West nach Ost über den Himmel führt. Diese Bewegung lässt sich am besten verfolgen, wenn die Mondbahn in der Nähe hellerer Sterne vorbeiführt. Besonders augenfällig wird sie bei einer Sternbedeckung, bei der ein Stern plötzlich hinter dem östlichen Mondrand verschwindet und erst nach einer ganzen Weile auf der gegenüberliegenden Seite wieder zum Vorschein kommt. Insgesamt gibt es rund tausend mit freiem Auge sichtbare Sterne, die vom Mond bedeckt werden können. Sie liegen alle in einem schmalen Band nördlich und südlich der Ekliptik, von der sie nicht mehr als 8° entfernt sind.

Eine Sternbedeckung ähnelt in mancher Hinsicht einer Sonnenfinsternis. Im parallelen Licht der sehr weit entfernten Sterne läuft der Mondschatte jedoch nicht kegelförmig zusammen, sondern hat überall den gleichen Durchmesser. Ebenso gibt es keine Unterscheidung zwischen Kern- und Halbschatten. Der Schatten, den der Mond im Licht eines Sterns wirft, ist so groß wie der Mond selbst ($1/4$ Erddurchmesser) und kann die Erde deshalb nie völlig abdecken. Dementsprechend lässt sich eine bestimmte Sternbedeckung immer nur von Teilen der Erde aus beobachten.

Durch das Fehlen einer Atmosphäre um den Mond wird der Stern vom Mond nicht allmählich, sondern schlagartig verdeckt und ebenso schlagartig wieder freigegeben. Die Ein- und Austrittszeiten lassen sich daher gut messen und erlauben so präzise Rückschlüsse auf die Position des Mondes. Die Beobachtung von Sternbedeckungen lieferte deshalb lange Zeit wichtige Informationen zur Verbesserung unseres Wissens über die Mondbewegung und die Drehung der Erde.

Für die Beobachtungsplanung werden die wichtigsten Daten von Sternbedeckungen in verschiedenen astronomischen Jahrbüchern und Zeitschriften veröffentlicht. Die Angaben beziehen sich dabei immer nur auf ausgewählte, zentral gelegene Orte. Mit Hilfe der sogenannten Stationskoeffizienten können die Ein- und Austrittszeiten aber auf beliebige andere Beobachtungsstationen umgerechnet werden. Die Berechnung solcher Vorhersagedaten erfordert im allgemeinen keine allzu hohe Genaugkeit, so daß einzelne Vereinfachungen gegenüber einer strengen Rechnung erlaubt sind. Beispielsweise kann das unregelmäßige Profil des Mondrands unberücksichtigt bleiben. Aufwendig wird die Vorhersage von Sternbedeckungen allerdings dadurch, daß zunächst für eine sehr große Anzahl von Sternen überprüft werden muß, ob eine Bedeckung in einem vorgegebenen Zeitraum überhaupt stattfindet.

Das Programm **Occult** berechnet mögliche Bedeckungen für eine beliebige Auswahl von Sternen in Abschnitten von je einem Tag. Innerhalb dieser Zeitspan-

Tabelle 10.1. Koordinaten zur Epoche J2000 und Eigenbewegung in 100 Jahren für verschiedene Plejadensterne auf Grundlage des PPM Katalogs. Angegeben sind zusätzlich die Nummer der Sterne im Zodiakalkatalog (ZC), die visuelle Helligkeit und die gebräuchlichen Namen.

ZC	$\alpha_{\text{J}2000}$	μ_α	$\delta_{\text{J}2000}$	μ_δ	m_{vis}	Name	
	h m s	s	° ' "	"	m		
536	3 44 48.180	+0.06	+24 17 21.44	-5.1	5.4	Celaeno	16 Tau
537	3 44 52.532	+0.14	+24 6 48.01	-4.6	3.8	Electra	17 Tau
539	3 45 12.469	+0.07	+24 28 1.36	-5.8	4.4	Taygeta	19 Tau
541	3 45 49.566	+0.05	+24 22 3.63	-5.0	4.0	Maia	20 Tau
542	3 45 54.423	+0.04	+24 33 16.02	-4.6	5.9	Asterope	21 Tau
545	3 46 19.537	+0.08	+23 56 53.42	-5.7	4.3	Merope	23 Tau
552	3 47 29.073	+0.14	+24 6 18.38	-4.6	3.0	Alcyone	η Tau
560	3 49 9.739	+0.13	+24 3 12.24	-4.7	3.8	Atlas	27 Tau
561	3 49 11.181	+0.09	+24 8 12.58	-4.3	5.2	Pleione	28 Tau

ne ist maximal eine Begegnung des Monds mit einem bestimmten Stern möglich. Der kleinste gegenseitige Abstand ist ungefähr dann erreicht, wenn die geozentrische Rektaszension des Mondes den gleichen Wert hat wie die Rektaszension des Sterns. Wann dies der Fall ist, lässt sich mit einer einfachen Iteration bestimmen. Stellt man dabei fest, daß die Deklinationsdifferenz ebenfalls nicht allzu groß ist, dann ist eine Sternbedeckung wenigstens für einen Teil der Erde möglich. Anschließend kann man die Bewegung des Mondschattens um den Zeitpunkt der Konjunktion herum im einzelnen untersuchen und so feststellen, ob es auch für den vorgegebenen Beobachtungsort zu einer Bedeckung kommt.

10.1 Scheinbare Sternkoordinaten

Die Koordinaten eines Sterns, für den man eine Bedeckung berechnen möchte, kann man in einer Reihe von Katalogen nachschlagen. Beispiele hierfür sind der PPM-Katalog (Position and Proper Motion Catalogue), der SAO-Katalog des Smithsonian Astrophysical Observatory oder der Zodiakalkatalog (Catalog of 3539 Zodiacaal Stars for the Equinox 1950.0), der speziell für die Auswertung von Sternbedeckungen aufgestellt wurde. Ein kleiner Auszug eines Sternkatalogs mit den wichtigsten Daten verschiedener Plejadensterne ist in Tabelle 10.1 wiedergegeben. Neben der Rektaszension und Deklination für die Epoche J2000 ist auch die jeweilige Eigenbewegung der Sterne in hundert Jahren aufgeführt. Wie man sieht, führt die räumliche Bewegung der Sterne gegenüber dem Sonnensystem im Lauf der Zeit zu durchaus merklichen Änderungen ihrer Koordinaten. Es empfiehlt sich deshalb, die Positionen (α_0, δ_0) zur Katalogepoch t_0 zunächst mit Hilfe der Angaben über die Eigenbewegung (μ_α, μ_δ) auf den ungefähren Zeitpunkt t der Sternbedeckung umzurechnen:

$$\begin{aligned}\alpha(t) &= \alpha_0 + \mu_\alpha(t - t_0) \\ \delta(t) &= \delta_0 + \mu_\delta(t - t_0) .\end{aligned}\tag{10.1}$$

Die Positionen in den genannten Katalogen können jedoch auch nach dieser Korrektur nicht unmittelbar für die Berechnung von Sternbedeckungen verwendet werden. Für den Vergleich mit der Mondbahn benötigt man die sogenannten *scheinbaren* Koordinaten (apparent places) des betrachteten Sterns. Diese beziehen sich auf die durch Präzession und Nutation bestimmte aktuelle Lage von Frühlingspunkt und Himmelsäquator, also auf das wahre Äquinoktium des Datums. Das Koordinatensystem ist dabei immer parallel zur momentanen Rotationsachse der Erde ausgerichtet. Demgegenüber beziehen sich die Positionen der verschiedenen Sternverzeichnisse auf ein Koordinatensystem, das in Richtung des mittleren Äquators und Frühlingspunktes einer festen Epoche ausgerichtet ist. Die aktuellen Kataloge beziehen sich meist auf die Epoche und das Äquinoktium des Jahres 2000, einer Reihe von Katalogen liegt aber noch das früher gebräuchliche Äquinoktium 1950 zugrunde.

Die Präzessionsmatrix, die die mittlere langfristige Wanderung des Frühlingspunktes beschreibt, kann über die Gleichungen (2.14) und (2.15) berechnet werden. Mit ihrer Hilfe lässt sich der Richtungsvektor

$$\mathbf{e} = \begin{pmatrix} \cos \delta \cos \alpha \\ \cos \delta \sin \alpha \\ \sin \delta \end{pmatrix} \quad (10.2)$$

der Sternkoordinaten dann vom mittleren Äquinoktium der Katalogepoche ins mittlere Äquinoktium des jeweiligen Datums transformieren.

Durch die wechselnden Anziehungskräfte von Sonne und Mond ist der Präzession der Erdachse eine zusätzliche periodische Schwankung – die Nutation – überlagert. Sie wird durch die beiden Winkel $\Delta\epsilon$ und $\Delta\psi$ beschrieben, die maximal $9''$ bzw. $17''$ betragen (vgl. (6.14)). Um Werte dieser Größenordnung unterscheiden sich auch die mittleren Koordinaten des Sterns von seinen wahren Koordinaten. Zur gegenseitigen Umrechnung dient die Transformationsmatrix (6.15).

Eine weitere notwendige Korrektur der Sternkoordinaten betrifft die *Aberration*, die durch die endliche Lichtgeschwindigkeit verursacht wird. Ein Beobachter, der sich mit der Erde um die Sonne bewegt, sieht das Licht des Sterns aus einer etwas anderen Richtung eintreffen als ein relativ zur Sonne ruhender Beobachter. Die beobachtete Sternposition \mathbf{e}' lässt sich ausreichend genau bestimmen, indem man zum Vektor \mathbf{e} der rechtwinkeligen Sternkoordinaten das Verhältnis \mathbf{v}_\oplus/c von Erdgeschwindigkeit und Lichtgeschwindigkeit addiert und das erhaltene Ergebnis wieder normiert:

$$\mathbf{e}' = \frac{\mathbf{e} + \mathbf{v}_\oplus/c}{|\mathbf{e} + \mathbf{v}_\oplus/c|} \quad (10.3)$$

Aus den so korrigierten kartesischen Koordinaten folgen dann die Rektaszension und Deklination des Sterns unter Berücksichtigung der Aberration.

Die vollständige Korrektur der aus einem Sternkatalog entnommenen Koordinaten unter Berücksichtigung der Eigenbewegung, der Präzession und Nutation sowie der Aberration kann damit wie im folgenden Beispiel programmiert werden:

```

// Eigenbewegung
double RA = RA_Cat + mu_RA *(T-T_CatEpoch);
double Dec = Dec_Cat + mu_Dec*(T-T_CatEpoch);
// Transformation vom mittleren Aequator und Fruehlingspunkt des Katalogs
// zum wahren Aequator und Fruehlingspunkt des Datums
Mat3D PN = NutMatrix(T)*PrecMatrix_Equ(T_CatEquinox,T);
// Praezession und Nutation
Vec3D e = PN * Vec3D(Polar(RA, Dec));
// Geschwindigkeitsvektor der Erde bezogen auf den mittleren Aequator
// und Aequinoktium des Datums in Einheiten der Lichtgeschwindigkeit
Vec3D v_Earth = Ecl2EquMatrix(T)*KepVelocity(Earth,T)/c_light;
// Aberration
e = e + v_Earth; e = e / Norm(e);

```

Zur Berechnung der Präzessions- und Nutationsmatrix werden die bekannten Funktionen PrecMatrix_Equ und NutMatrix verwendet. Die heliozentrische Geschwindigkeit der Erde wird mit Hilfe von KepVelocity auf Grundlage einer genähernten Keplerbahn berechnet und anschließend von der Ekliptik ins Äquatorsystem transformiert. Als Ergebnis erhält man den Vektor der scheinbaren Sternkoordinaten.

Zur Verwaltung des Sternkatalogs, auf dessen Grundlage in Occult Bedeckungen durch den Mond vorhergesagt werden, dient eine eigene Klasse Star:

```

class Star {
public:
    static void SetCatEquinox( double T );
    static void SetCatEpoch( double T );
    static void SetEpoch( double T );
    Vec3D Apparent();           // Berechne scheinbaren Ort
    inline char* Name() { return name; };
    inline double Mag() { return mag; };
    friend istream& operator >> (istream& is, Star& aStar);
private:
    double RA_Cat, Dec_Cat;     // Rektaszension/Deklination nach Katalog [rad]
    double mu_RA, mu_Dec;       // Eigenbewegung in Rekt./Dekl. [rad/cy]
    double mag;                 // Visuelle Helligkeit in [mag]
    char name[12];              // Name
    static double T_CatEquinox; // Aequinoktium des Katalogs
    static double T_CatEpoch;   // Epoche des Katalogs
    static double T_Epoch;      // Aktuelle Epoche
    static Mat3D PN;            // Matrix fuer Praezession und Nutation
    static Vec3D v_Earth;        // Geschwindigkeitsvektor der Erde
};

```

Die Klasse stellt einen Eingabeoperator `>>` zur Verfügung, mit dessen Hilfe die Koordinaten, die Eigenbewegung, die Helligkeit und der Name eines Sterns aus einer Textdatei gelesen werden und in entsprechenden Attributen gespeichert werden kann. Allen Objekten der Klasse gemeinsam sind die als `static` deklarierten Attribute zur Speicherung der Katalogepoche und des Katalogäquinoktiums, die über entsprechende Methoden

```

void Star::SetCatEquinox (double T) { T_CatEquinox = T; }
void Star::SetCatEpoch (double T) { T_CatEpoch = T; }

```

gesetzt werden können. In ähnlicher Weise kann über die Methode

In beiden Fällen wird ferner ($t_2, \Delta\alpha_2$) durch ($t', \Delta\alpha'$) ersetzt. Diese als *Pegasus-Verfahren* bezeichnete Variante der *regula falsi* stellt sicher, daß kein Endpunkt des Suchintervalls im Verlauf der Iteration festgehalten wird und erreicht so eine deutlich höhere Konvergenzordnung.

```
-----
// Pegasus: Nullstellenbestimmung nach dem Pegasus-Verfahren
// PegasusFunct Zeiger auf die zu untersuchende Funktion
// LowerBound Untere Grenze des Suchintervalls
// UpperBound Obere Grenze des Suchintervalls
// Accuracy Geforderte Genauigkeit der Nullstelle
// Root Nullstelle (gueltig, falls Success = true)
// Success Angabe, ob eine Nullstelle gefunden wurde
// Bemerkungen:
// Pegasus nimmt an, dass die zu findende Nullstelle im Intervall
// [LowerBound, UpperBound] eingeschlossen ist. Die Funktionswerte an den
// Intervallgrenzen muessen daher unterschiedliche Vorzeichen aufweisen.
// -----
void Pegasus ( PegasusFunct f,
                double LowerBound, double UpperBound, double Accuracy,
                double& Root, bool& Success )
{
    const int MaxIterat = 30;
    double x1 = LowerBound; double f1 = f(x1);
    double x2 = UpperBound; double f2 = f(x2);
    double x3 = 0.0;           double f3 = 0.0;
    int Iterat = 0;
    // Initialisierung
    Success = false;
    Root     = x1;
    // Iteration
    if ( f1 * f2 < 0.0 )
        do
    {
        // Naehlerungswert fuer die Nullstelle durch Interpolation bestimmen
        x3 = x2 - f2/( (f2-f1)/(x2-x1) ); f3 = f(x3);
        // Ersetze (x1,f2) und (x2,f2) so durch neue Werte, dass die Nullstelle
        // im Intervall [x1,x2] bleibt
        if ( f3 * f2 <= 0.0 ) {
            // Nullstelle in [x2,x3]
            x1 = x2; f1 = f2; // Ersetze (x1,f1) durch (x2,f2)
            x2 = x3; f2 = f3; // Ersetze (x2,f2) durch (x3,f3)
        }
        else {
            // Nullstelle in [x1,x3]
            f1 = f1 * f2/(f2+f3); // Ersetze (x1,f1) durch (x1,f1')
            x2 = x3; f2 = f3;      // Ersetze (x2,f2) durch (x3,f3)
        }
        if (fabs(f1) < fabs(f2)) Root=x1; else Root=x2;
        Success = (fabs(x2-x1) <= Accuracy);
        Iterat++;
    }
    while ( !Success && (Iterat<MaxIterat) );
}
```

Mit der hier vorgestellten Funktion `Pegasus` kann allgemein die Nullstelle einer skalaren Funktion $f(x)$ vom Typ

```
// Funktionsprototyp fuer die Funktion Pegasus: double f(double x);
typedef double (*PegasusFunct) (double x);
```

bestimmt werden.

Bei der Suche nach der Konjunktionszeit müssen ebenso wie im weiteren Verlauf der Rechnung eine ganze Reihe von Mondpositionen bestimmt werden. Es lohnt sich daher, auf die schon früher besprochene Entwicklung der Mondkoordinaten nach Tschebyscheff-Polynomen zurückzugreifen. Diese erlaubt es, aus wenigen, exakt berechneten Positionen eine unkomplizierte Darstellung der Mondbahn durch ein Polynom zu gewinnen, das dann mit wenigen Operationen für beliebig viele Zeitpunkte ausgewertet werden kann. Der maximale Grad (z.B. zehn) der verwendeten Polynome wird dabei so gewählt, daß die Mondkoordinaten über einen Zeitraum von rund einem Tag ohne Genauigkeitsverlust dargestellt werden können. Wie in Kap. 8 beschrieben, genügt es hierzu ein Objekt der Klasse `Cheb3D` zur Approximation der Funktion `MoonEqu` zu deklarieren. Anschließend kann durch Aufruf der Methode `Value(T)` der Vektor der äquatorialen Mondkoordinaten zu einer vorgegebenen Zeit T bestimmt werden. Die Rektaszension erhält man hieraus über den Zugriffsoperator `[phi]`. Die so bestimmten Koordinaten beziehen sich auf das wahre Äquinoktium des Datums, also auf die durch Präzession und Nutation bestimmte aktuelle Lage von Frühlingspunkt und Himmelsäquator.

Da `Pegasus` als Argument eine Funktion erwartet, die nur von einem Parameter abhängt, ist es erforderlich, die Tschebyscheff-Approximation des Mondes ebenso wie die Rektaszension des Sterns als globale Größen zu deklarieren. Die Zielfunktion $\Delta\alpha(T)$ zur Nullstellensuche kann dann wie folgt implementiert werden:

```
const double Interval=1.0/36525.0;      // 1d [cy]
Cheb3D ChebMoonEqu(MoonEqu, 10, Interval);
double RA_Star;

//-----
// RA_Diff: Zielfunktion fuer die Iteration der Konjunktionszeit
//   T      Zeit in Julianischen Jahrhunderten ET seit J2000
//   <return> Differenz der Rekt. von Mond und Stern in [rad]
//-----
double RA_Diff (double T)
{
    Vec3D r_Moon = ChebMoonEqu.Value(T);  // Mondposition zur Zeit T
    return Modulo ( r_Moon[phi] - RA_Star + pi, pi2 ) - pi;
}
```

Für einen gegebenen Stern kann damit der genaue Zeitpunkt der Konjunktion ermittelt werden, wenn im betrachteten Zeitintervall überhaupt ein Vorzeichenwechsel von $\Delta\alpha(T)$ stattfindet.

Ausgehend von den Mondkoordinaten zur Konjunktionszeit lässt sich nun die Möglichkeit einer Sternbedeckung überprüfen. Wie man aus Abb. 10.2 entnehmen kann, trifft der vom Mond erzeugte Schatten die Erde nur, wenn sich die

```
void Star::SetEpoch (double T) {
    T_Epoch = T; // Speichere Epoche
    PN = NutMatrix(T)*PrecMatrix_Equ(T_CatEquinox, T);
    v_Earth = Ecl2EquMatrix(T)*KepVelocity(Earth,T)/c_light;
}
```

die aktuelle Epoche festgelegt werden. Dabei werden gleichzeitig die – für alle Sterne gleichen – Werte der Präzessions-/Nutationsmatrix und der heliozentrischen Geschwindigkeit der Erde ermittelt. Die genannten Größen dienen anschließend zur Berechnung der scheinbaren Koordinaten innerhalb der Methode `Apparent`:

```
Vec3D Star::Apparent() {
    double RA = RA_Cat + mu_RA*(T_Epoch-T_CatEpoch);
    double Dec = Dec_Cat + mu_Dec*(T_Epoch-T_CatEpoch);
    Vec3D e = PN * Vec3D(Polar(RA, Dec));
    e = e + v_Earth; e = e / Norm(e);
    return e;
};
```

Zur Speicherung des gesamten Sternkatalogs wird in `Occult` ein Feld von Objekten der Klasse `Star` verwendet, das in der Funktion `ReadCatalogue` dynamisch allokiert und belegt wird:

```
-----
// ReadCatalogue: Liest den Sternkatalog aus einer Datei ein
//   Filename      Name der einzulesenden Katalogdatei
//   Stars         Feld mit eingelesenen Objekten der Klasse Star
//   StarsRead     Anzahl eingelesener Objekte
-----
void ReadCatalogue (char* Filename, Star*& Stars, int& StarsRead)
{
    // Variablen
    ifstream inp;
    double Y_Epoch, Y_Eqx;
    int i;
    inp.open(Filename);

    // Epoche und Aequinoktium lesen und setzen
    inp >> Y_Epoch >> Y_Eqx;
    Star::SetCatEpoch((Y_Epoch-2000.0)/100.0);
    Star::SetCatEquinox((Y_Eqx-2000.0)/100.0);

    // Sterndaten einlesen
    inp >> StarsRead; inp.ignore(81, '\n');
    Stars = new Star[StarsRead]; // Speicherplatz fuer Sterndaten anfordern
    if (Stars != NULL) {
        for (i=0; i<StarsRead; i++) inp >> Stars[i];
    }
    else
        cerr << "Unzureichender Speicher zum Einlesen des Sternkatalogs." << endl;
    inp.close();
    cout << endl << " " << StarsRead
        << " Sterne aus dem Katalog eingelesen" << endl << endl;
}
```

10.2 Die geozentrische Konjunktion

Für die Vorhersage von Sternbedeckungen müssen aus einer großen Zahl ekliptiknaher Sterne diejenigen ausgewählt werden, die in einem vorgegebenen Zeitraum vom Mond bedeckt werden können. Zu diesem Zweck kann man zunächst für jeden Stern den Zeitpunkt bestimmen, in dem seine Rektaszension α_* mit der geozentrischen Rektaszension α_M des Mondes übereinstimmt. Ein Vergleich der Deklinationen von Mond und Stern zum Zeitpunkt dieser Konjunktion lässt dann erkennen, ob der Stern zumindest für einen Teil der Erde bedeckt wird. Der Zeitpunkt der Konjunktion bildet darüber hinaus den Ausgangspunkt für die genaue Berechnung möglicher Bedeckungen, die in den späteren Abschnitten behandelt wird.

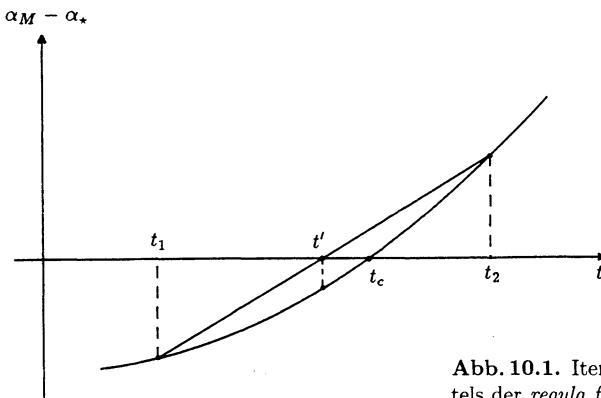


Abb. 10.1. Iteration der Konjunktionszeit mittels der *regula falsi*

Zwischen zwei Zeitpunkten t_1 und t_2 variiert die Rektaszension des Mondes zwischen $\alpha_M(t_1)$ und $\alpha_M(t_2)$. Liegt die Rektaszension α_* eines Sterns zwischen diesen beiden Werten, dann kann man die Konjunktionszeit t_c durch eine lineare Interpolation bestimmen (Abb. 10.1). Da die Rektaszension des Mondes sehr gleichmäßig anwächst, gilt in guter Näherung

$$t_c \approx t' = t_2 - \Delta\alpha_2 \cdot \frac{t_2 - t_1}{\Delta\alpha_2 - \Delta\alpha_1} .$$

mit $\Delta\alpha_i = \alpha_M(t_i) - \alpha_*$. Je nachdem, ob die Rektaszension des Mondes zur Zeit t' größer oder kleiner als die des Sterns ist, ersetzt man nun t_1 oder t_2 durch t' . Es gilt dann wieder $\alpha_M(t_1) \leq \alpha_* \leq \alpha_M(t_2)$, so daß man die Konjunktionszeit auf die gleiche Weise weiter verbessern kann. Wegen der gleichförmigen Bewegung des Mondes führt dieses als *regula falsi* bezeichnete Verfahren im allgemeinen zügig zum gewünschten Ergebnis. Durch eine kleine Modifikation lässt sich die Konvergenz jedoch noch deutlich verbessern. Dazu werden nach Berechnung von t' und $\Delta\alpha' = \alpha_M(t') - \alpha_*$ die Startwerte für die jeweils nachfolgende Iteration wie folgt gewählt:

- falls $\Delta\alpha'\Delta\alpha_2 \leq 0$, ersetze $(t_1, \Delta\alpha_1)$ durch $(t_2, \Delta\alpha_2)$;
- falls $\Delta\alpha'\Delta\alpha_2 > 0$, ersetze $(t_1, \Delta\alpha_1)$ durch $(t_1, \Delta\alpha_1\Delta\alpha_2 / (\Delta\alpha_2 + \Delta\alpha'))$.

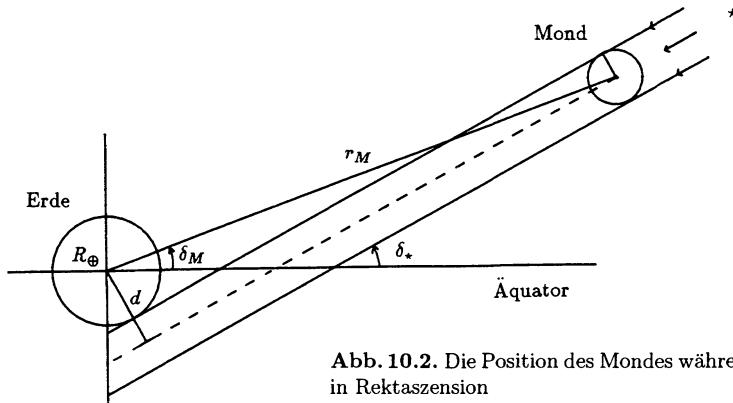


Abb. 10.2. Die Position des Mondes während der Konjunktion in Rektaszension

Deklination des Mondes nicht allzu sehr von der des Sterns unterscheidet. Die Entfernung d der Schattenachse vom Erdmittelpunkt darf dabei nicht größer sein, als die Summe aus Mondradius R_M und Erdradius R_\oplus :

$$d < R_\oplus + R_M \approx 1.3R_\oplus .$$

Während der Konjunktion in Rektaszension muß dazu die Bedingung

$$d(t_c) = r_M \cdot |\sin(\delta_M - \delta_*)| = |\mathbf{r}_M - (\mathbf{e} \cdot \mathbf{r}_M)\mathbf{e}| < R_\oplus + R_M \approx 1.3R_\oplus$$

erfüllt sein. Aufgrund der Neigung der Mondbahn gegen den Himmelsäquator kann $d(t_c)$ allerdings um bis zu 0.2 Erdradien größer sein, als der kleinste erreichte Abstand des Mondschatzens vom Erdmittelpunkt. Zur Auswahl möglicher Sternbedeckungen wird deshalb im folgenden die Unterscheidung

$$|\mathbf{r}_M - (\mathbf{e} \cdot \mathbf{r}_M)\mathbf{e}| \left\{ \begin{array}{l} < \\ > \end{array} \right\} 1.5R_\oplus \Rightarrow \left\{ \begin{array}{l} \text{Bedeckung möglich} \\ \text{keine Bedeckung möglich} \end{array} \right\} \quad (10.4)$$

verwendet.

Der erste Teil des Sternbedeckungsprogrammes kann damit formuliert werden. *Conjunct* prüft zuerst, ob die Rektaszension des Sterns überhaupt im Bereich der Rektaszensionswerte liegt, die der Mond zwischen zwei vorgegebenen Zeiten T_1 und T_2 durchläuft. Ist dies der Fall, dann wird der Zeitpunkt der Konjunktion gesucht und getestet, ob der Mondschatzen auf die Erde trifft. Wenn alle Bedingungen erfüllt sind, wird die Konjunktionszeit an das aufrufende Programm zurückgegeben.

```
-----
// Conjunct: prueft, ob und wann in einem Zeitintervall [T1,T2] eine
// Konjunktion von Mond und Stern stattfindet, bei der der
// Mondschatzen die Erde trifft.
// T1      Anfang des Suchintervalls
// T2      Ende des Suchintervalls
// RA1    Rektaszension des Mondes zur Zeit T1 in [rad]
// RA2    Rektaszension des Mondes zur Zeit T2 in [rad]
// e      Richtungsvektor zum Stern (aequatoriale Koordinaten)
```

```

// Conj      zeigt an, ob eine Bedeckung zumindest wahrscheinlich ist
// T_Conj    Zeit der Konjunktion
//-----
void Conjunction ( double T1, double T2, double RA1, double RA2, Vec3D& e,
                   bool& Conj, double& T_Conj )
{
    const double eps = Rad*1.0e-4;      // Gewuenschte Genauigkeit in RA [rad]
    Vec3D r_Moon;
    T_Conj = 0.0;                      // Vorgabewert
    bool Success = false;              // Abbilden auf [RA1,RA1+2*pi[
    if ( RA2<RA1 ) RA2+=pi2;          // Abbilden auf [RA1,RA1+2*pi[
    RA_Star = e[phi];
    if ( RA_Star<RA1 ) RA_Star+=pi2;   // Abbilden auf [RA1,RA1+2*pi[
    Conj = ( (RA1<=RA_Star) && (RA_Star<=RA2) );
    if ( Conj ) {
        Pegasus(RA_Diff, T1, T2, eps, T_Conj, Success);
        r_Moon = ChebMoonEqu.Value(T_Conj);
        Conj = ( Norm( r_Moon-e*Dot(r_Moon,e) ) < 1.5*R_Earth );
    }
}

```

10.3 Die Fundamentalebene

Im parallelen Licht des Sterns wirft der Mond einen Schatten, der sich als langer Zylinder vom Mond zur Erde erstreckt. Auf jeder Ebene, die senkrecht zur Schattenachse liegt, zeichnet sich der Schatten als ein Kreis ab, der den gleichen Durchmesser hat, wie der Mond selbst. Diejenige Ebene, die zusätzlich durch den Erdmittelpunkt geht, bezeichnet man als *Fundamentalebene*. In der Projektion auf die Fundamentalebene lässt sich der Eintritt des Beobachters in den Mondschatzen besonders gut beschreiben.

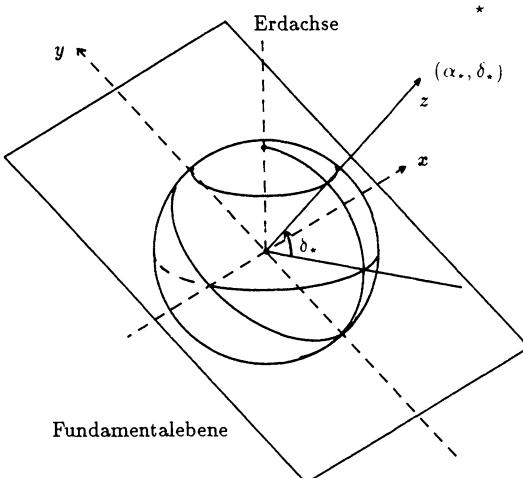


Abb. 10.3. Die Fundamentalebene

Legt man wie in Abb. 10.3 ein rechtwinkeliges Koordinatensystem so in den Erdmittelpunkt, daß die z-Achse in Richtung des Sterns mit den Koordinaten

(α_*, δ_*) weist und daß die x -Achse in der Äquatorebene der Erde liegt, dann spannen die x - und die y -Achse gerade die Fundamentalebene auf. Die zugehörigen Einheitsvektoren berechnen sich zu

$$\mathbf{e}_x = \frac{\mathbf{e}_3 \times \mathbf{e}}{|\mathbf{e}_3 \times \mathbf{e}|} \quad \mathbf{e}_y = \mathbf{e} \times \mathbf{e}_x \quad (10.5)$$

wobei $\mathbf{e} = \mathbf{e}_z$ den Richtungsvektor des Sternortes (vgl. (10.2)) und $\mathbf{e}_3 = (0, 0, 1)^T$ den Einheitsvektor in Richtung des Himmelsnordpols bezeichnet.

Durch Projektion des äquatorialen Ortsvektors \mathbf{r}_M des Mondes auf die genannten Basisvektoren erhält man die Koordinaten

$$\begin{aligned} x_M &= \mathbf{e}_x \cdot \mathbf{r}_M \\ y_M &= \mathbf{e}_y \cdot \mathbf{r}_M \end{aligned} \quad (10.6)$$

der Mondschattemitte auf der Fundamentalebene. Ganz entsprechend ergibt sich aus den äquatorialen Koordinaten \mathbf{r}_B des Beobachtungsortes der Punkt (x_B, y_B) der Fundamentalebene, auf den ein Lichtstrahl vom Stern den Beobachter projiziert. Die Deklination des Beobachtungsortes beschreibt dabei den Winkel zwischen dem geozentrischen Ortsvektor und der Ebene des Erd- und Himmelsäquators. Sie wird in diesem Zusammenhang auch als *geozentrische Breite* φ' bezeichnet, sollte aber nicht mit der *geographischen Breite* φ verwechselt werden. Diese weitaus häufiger verwendete Größe gibt den Winkel zwischen der Erdachse und dem lokalen Horizont an, ist also zum Beispiel ein Maß für die Höhe des Polarsterns über dem Horizont. Die Ursache für den Unterschied von φ und φ' liegt in der Abplattung der Erde (vgl. Abb. 9.5). Aufgrund ihrer täglichen Drehung hat die Erde nicht die Form einer Kugel, sondern die eines Rotationsellipsoids. Während der Erdradius am Äquator $r_{\oplus} = 6378.14$ km beträgt, ist die Entfernung der Pole vom Erdmittelpunkt rund 20 km kleiner. Das Verhältnis dieser Differenz zum Erdradius wird als Abplattung¹ $f \approx 1/298$ bezeichnet. Im allgemeinen ist die geozentrische Breite eines Ortes nicht direkt verfügbar und muß deshalb aus der geographischen Breite berechnet werden. Diese kann ihrerseits aus gängigen Kartenwerken und Verzeichnissen entnommen werden.

Die Funktion **Site** ermittelt daraus über (9.9) die kartesischen Koordinaten

$$\mathbf{r}_B^G = \begin{pmatrix} r \cos \lambda \cos \varphi' \\ r \sin \lambda \cos \varphi' \\ r \sin \varphi' \end{pmatrix} = \frac{R_{\oplus}}{\sqrt{1 - e^2 \sin^2 \varphi}} \begin{pmatrix} \cos \lambda \cos \varphi \\ \sin \lambda \cos \varphi \\ (1 - e^2) \sin \varphi \end{pmatrix} \quad (10.7)$$

des Beobachtungsortes bezogen auf den Erdäquator und den Meridian von Greenwich. Bei Kenntnis der Greenwich-Sternzeit $\Theta_0(t)$ folgt hieraus der Ortsvektor

$$\mathbf{r}_B = \mathbf{R}_z(-\Theta_0(t)) \mathbf{r}_B^G \quad (10.8)$$

bezogen auf Äquator und Frühlingspunkt. Die Sternzeit selbst kann mit Hilfe der Funktion **GMST** aus Abschn. 3.3 berechnet werden.

¹ f bezeichnet traditionell sowohl die Abplattung als auch eine der beiden Fundamentalebenenkoordinaten. Eine Verwechslung sollte aber weitgehend ausgeschlossen sein.

```

//-----
// Site: Berechnet die geozentrische Position eines Orts an der Erdoberflaeche
// lambda Geographische Laenge (ostwaerts positiv) in [rad]
// phi Geographische Breite in [rad]
// <return> geozentrische Position in [km]
//-----

Vec3D Site (double lambda, double phi)
{
    // Constants
    const double f      = 1.0/298.257;    // Abplattung der Erde
    const double e_sqr = f*(2.0-f);        // Quadrat der Exzentrizitaet
    const double cos_phi = cos(phi);       // (Ko)sinus der geographischen Breite
    const double sin_phi = sin(phi);
    // Variablen
    double N = R_Earth / sqrt (1.0-e_sqr*(sin_phi*sin_phi));
    // Kartesischer Ortsvektor [km]
    return Vec3D ( N*cos_phi*cos(lambda),
                  N*cos_phi*sin(lambda),
                  (1.0-e_sqr)*N*sin_phi );
}

```

An dieser Stelle ist es notwendig, noch einmal auf die verschiedenen astronomischen Zeitzählungen einzugehen. Bei der Berechnung der Mondpositionen wurde bisher stillschweigend davon ausgegangen, daß alle Zeiten in Form der *Ephemeridenzeit* ET (= dynamische Zeit TDB/TDT) gegeben sind. Dieses im physikalischen Sinne gleichförmige Zeitmaß wird überall dort verwendet, wo die Bahnen von Himmelskörpern berechnet werden. Hierauf ist letztlich ja auch der Name Ephemeridenzeit zurückzuführen. Demgegenüber erfordert die Berechnung der Sternzeit die Kenntnis der *Weltzeit* UT, die (bis auf unterschiedliche Zeitzonen) im wesentlichen unserer gewohnten Uhrzeit entspricht. Der Unterschied von ET und UT liegt derzeit bei rund einer Minute und kann aus Tabelle 3.1 entnommen werden. Weiterhin ist in Abschn. 9.3 eine Funktion ETminUT angegeben, die für den Zeitraum zwischen 1825 und 2000 die gesuchten Werte aus einem Näherungs polynom berechnet. Für spätere Jahre sind aktuelle Angaben in verschiedenen astronomischen Jahrbüchern zu finden. Da sich die Differenz zwischen Welt- und Ephemeridenzeit nur langsam verändert, muß sie nur einmal zu Programmbeginn ermittelt werden.

10.4 Ein- und Austritt

Der Verlauf der Sternbedeckung hängt von den Differenzen

$$\begin{aligned} f &= x_M - x_B \\ g &= y_M - y_B \end{aligned}$$

der Koordinaten von Mond und Beobachter in der Fundamentalebene ab. Ist der Abstand $\sqrt{f^2 + g^2}$ des Beobachters von der Mitte des Mondsattens kleiner als der Radius R_M des Mondes, dann wird der Stern für den ausgewählten Beobachtungsort vom Mond bedeckt. Die Ein- und Austrittszeit erhält man demnach aus der Bedingung

$$f(t)^2 + g(t)^2 = R_M^2 = k^2 \cdot R_\oplus^2 . \quad (10.9)$$

Darin ist

$$k = R_M/R_\oplus = 0.2725$$

das Verhältnis zwischen Mondradius und Erdradius.

In seiner Bahn um die Erde legt der Mond eine Strecke von der Größe des Erddurchmessers in rund vier Stunden zurück. In dieser Zeit wandert auch der Mondschatz während einer Sternbedeckung über die Erde. Bei der Suche der Kontaktzeiten genügt es daher, etwa zweieinhalb Stunden vor dem Zeitpunkt der Konjunktion zu beginnen. Um nun die beiden Zeiten zu finden, die die Bedingung (10.9) erfüllen, werden in Abständen von 15^m die Werte

$$s(t) = f^2(t) + g^2(t) - k^2 R_\oplus^2$$

berechnet. s ist anfangs positiv, da sich der Beobachtungsort noch außerhalb des Mondschatzens befindet. Wird s negativ, dann verschwindet der Stern für den Beobachter hinter dem Mondrand. Aus drei aufeinanderfolgenden Werten s_- , s_0 und s_+ kann man nun eine Parabel bestimmen, die den Verlauf von $s(t)$ über eine halbe Stunde hinweg in guter Näherung wiedergibt. Hat s in dieser Zeit eine oder zwei Nullstellen, dann lassen sich diese als Lösungen einer quadratischen Gleichung bestimmen. Ansonsten wählt man die nächsten drei s -Werte und verfährt mit diesen in entsprechender Weise. Dieses Vorgehen entspricht genau der Bestimmung der Auf- und Untergangszeiten in Kap. 3. Dort wurde auch bereits die Prozedur Quad zur quadratischen Interpolation und Nullstellenbestimmung einer Funktion aus drei Punkten vorgestellt. Sie wird innerhalb der Funktion Examine zur Bestimmung der Ein- und Austrittszeit verwendet. Der vollständige Text dieser Routine ist am Ende des Kapitels wiedergegeben.

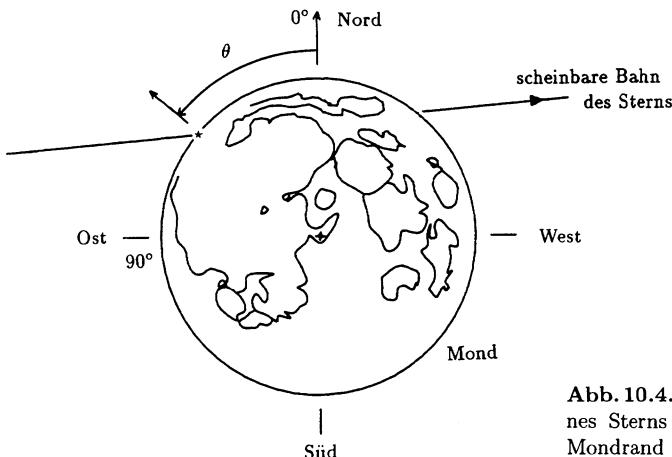


Abb. 10.4. Der Positionswinkel eines Sterns beim Kontakt mit dem Mondrand

Neben den Kontaktzeiten werden bei der Vorhersage von Sternbedeckungen einige weitere Größen angegeben, die für den Beobachter von besonderem Interesse sind: die Positionswinkel des Ein- und Austritts und die Stationskoeffizienten. Die Definition des Positionswinkels ist in Abb. 10.4 dargestellt. Man versteht darunter

den Winkel zwischen der Nordrichtung und der Linie von der Mondmitte zum Ein- oder Austrittspunkt des Sterns am Mondrand. Positionswinkel werden entgegen dem Uhrzeigersinn von 0° – 360° gemessen. Der Wert des Positionswinkels ϑ hängt nur von den Fundamentebenenkoordinaten f und g im Moment des Ein- oder Austritts ab und lässt sich daraus mit Hilfe des Zusammenhangs

$$\begin{aligned}\cos \vartheta &= -g / \sqrt{f^2 + g^2} \\ \sin \vartheta &= -f / \sqrt{f^2 + g^2} .\end{aligned}\quad (10.10)$$

berechnen.

Die Stationskoeffizienten sind Größen, die die ungefähre Abhängigkeit der Kontaktzeiten vom Beobachtungsort beschreiben. Man verwendet sie vor allem dazu, Kontaktzeiten, die für einen bestimmten Ort vorhergesagt wurden, auf benachbarte Orte zu übertragen. Der Koeffizient a gibt an, wie sich die Kontaktzeit t mit der geographischen Länge λ verändert, b ist der entsprechende Wert für die geographische Breite φ :

$$a = \frac{dt}{d\lambda} \quad b = \frac{dt}{d\varphi} .$$

Typische Werte der beiden Koeffizienten liegen bei etwa $1^\text{m}/^\circ$. Bei streifenden Sternbedeckungen, bei denen Ein- und Austritt dicht beieinander liegen, wachsen die Stationskoeffizienten jedoch deutlich an. Die Gleichungen zur Berechnung der Stationskoeffizienten enthalten im wesentlichen bereits bekannte Größen. Faßt man die Bedingung (10.9) als impliziten Zusammenhang zwischen der Kontaktzeit t und dem Beobachtungsort \mathbf{r} auf, dann erhält man durch Differentiation zunächst die Ableitung

$$\frac{dt}{d\mathbf{r}} = \frac{f \mathbf{e}_x + g \mathbf{e}_y}{f \dot{f} + g \dot{g}} \quad (10.11)$$

der Zeit des Ein- beziehungsweise Austritts nach den kartesischen Koordinaten des Beobachtungsortes. Hierin sind \dot{f} und \dot{g} die zeitlichen Ableitungen von f und g , die sich der Einfachheit halber durch die folgenden Differenzenquotienten ausdrücken lassen:

$$\dot{f}(t) \approx \frac{f(t + 0.25^\text{h}) - f(t)}{0.25^\text{h}} \quad \dot{g}(t) \approx \frac{g(t + 0.25^\text{h}) - g(t)}{0.25^\text{h}} .$$

Mit den Ableitungen

$$\frac{d\mathbf{r}}{d\lambda} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \mathbf{r} \quad \frac{d\mathbf{r}}{d\varphi} = \mathbf{r} \times \frac{d\mathbf{r}}{d\lambda} / \left| \frac{d\mathbf{r}}{d\lambda} \right| \quad (10.12)$$

des Ortsvektors nach der geographischen Länge und Breite ergeben sich dann die gesuchten Stationskoeffizienten zu

$$a = \frac{dt}{d\mathbf{r}} \frac{d\mathbf{r}}{d\lambda} \quad b = \frac{dt}{d\mathbf{r}} \frac{d\mathbf{r}}{d\varphi} . \quad (10.13)$$

Der Unterschied zwischen geographischer (φ) und geozentrischer Breite (φ') kann bei der Berechnung der Stationskoeffizienten vernachlässigt werden.

10.5 Das Programm OCCULT

Das Programm Occult berechnet Sternbedeckungen durch den Mond für einen gewünschten Suchzeitraum und einen vorzugebenden Ort. Berechnet werden der Zeitpunkt des Ein- und Austritts des Sterns am Mondrand, der zugehörige Positionswinkel, die Höhe des Mondes über dem Horizont zum Zeitpunkt des jeweiligen Kontakts und die Stationskoeffizienten, mit denen eine Umrechnung der Kontaktzeiten auf nahegelegene Orte möglich ist.

Die Koordinaten der Sterne, die auf Bedeckungen untersucht werden sollen, müssen in einer Datei mit dem standardmäßigen Namen `Occult.dat` bereitgestellt werden. Die erste Zeile dieser Datei enthält die Epoche und das Äquinoktium des zugrundeliegenden Katalogs (meist 1950.0 oder 2000.0) sowie die Anzahl der Sterne. Die beiden ersten Angaben dienen zur Berücksichtigung der Eigenbewegung und zur Umrechnung der Sternpositionen auf das aktuelle Äquinoktium. Anschließend sind für jeden Stern in einer gesonderten Zeile

- die Rektaszension (in $^{\text{h}} \text{ } ^{\text{m}} \text{ } ^{\text{s}}$),
 - die Eigenbewegung in Rektaszension in hundert Jahren (in $^{\text{s}}$),
 - die Deklination (in $^{\circ} \text{ } ' \text{ } ''$),
 - die Eigenbewegung in Deklination in hundert Jahren (in $''$) und
 - der Name

im Format

~~hh mm ss.ss ss.ss +dd mm ss.ss ss.ss mm.m cccccccccccccc~~

anzugeben. Die Länge des Namens ist dabei auf maximal 11 Buchstaben beschränkt, kann aber bei Bedarf durch Änderung der Klasse `Star` erhöht werden. Das folgende Beispiel für den Aufbau der Sterndatei enthält die Daten verschiedener Pleiadensterne (vgl. Tabelle 10.1).

2000.0	2000.0	9							
3 44 48.180	0.06	+24 17 21.44	-5.1	5.4	Celaeno				
3 44 52.532	0.14	+24 6 48.01	-4.6	3.8	Electra				
3 45 12.469	0.07	+24 28 1.36	-5.8	4.4	Taygeta				
3 45 49.566	0.05	+24 22 3.63	-5.0	4.0	Maia				
3 45 54.423	0.04	+24 33 16.02	-4.6	5.9	Asterope				
3 46 19.537	0.08	+23 56 53.42	-5.7	4.3	Merope				
3 47 29.073	0.14	+24 6 18.38	-4.6	3.0	Alcyone				
3 49 9.739	0.13	+24 3 12.24	-4.7	3.8	Atlas				
3 49 11.181	0.09	+24 8 12.58	-4.3	5.2	Pleione				

Nach dem Start des Programms werden die geographischen Koordinaten des Beobachtungsortes und die Zeitpunkte abgefragt, zwischen denen nach Sternbedeckungen gesucht werden soll. Der gesamte Zeitraum wird in Abschnitten von je einem Tag bearbeitet, für die die Mondkoordinaten zunächst durch Tschebyscheff-Polynome dargestellt werden. Für jeden Stern der Datei wird dann nach Berechnung der scheinbaren Koordinaten zur Tagesmitte in **Examine** die Möglichkeit einer Bedeckung untersucht. Die bereits beschriebene Funktion **Conjunct** bestimmt

zunächst den Zeitpunkt der Konjunktion von Mond und Stern und prüft die Entfernung der Schattenachse vom Erdmittelpunkt. Ergibt sich hieraus, daß eine Bedeckung prinzipiell stattfinden kann, dann ermittelt **Examine** für den gegebenen Beobachtungsort die genaue Ein- und Austrittszeit. Anschließend berechnet **Contact** für jeden der beiden Kontakte den zugehörigen Positionswinkel und die Stationskoeffizienten.

Von den so bestimmten Sternbedeckungen werden allerdings nur solche ausgetragen, die unter günstigen Sichtbarkeitsbedingungen stattfinden. Hierzu wird in **Contacts** die Höhe des Sterns und der Sonne über dem Horizont, die Phase des Mondes und die Beleuchtung des Mondrandes sowie die Helligkeit des Sterns nach folgenden Kriterien überprüft:

- Zum Zeitpunkt des Ein- oder Austritts muß der Stern mindestens 5° über dem Horizont stehen. Für Sterne heller als 1^m9 wird diese Grenze auf 2° reduziert.
- Die Helligkeit des Sterns muß mindestens 7^m5 betragen.
- Zum Zeitpunkt des Ein- oder Austritts muß die Sonne wenigstens 6° unter dem Horizont stehen (bürgerliche Dämmerung). Für Sterne heller als 5^m5 gilt eine reduzierte Grenze von 3° . Bedeckungen von Sternen der Größenklassen 2^m0 bis 4^m0 werden im gesamten Zeitraum von Sonnenuntergang bis Sonnenaufgang betrachtet, bei noch helleren Sternen auch Ereignisse, die tagsüber stattfinden.
- Für Phasenwinkel unter 12° , also rund 24 Stunden vor und nach Vollmond, gilt eine Grenzgröße von 3^m5 , bis 24° von 5^m5 und bis 36° von 6^m5 .
- Am hellen Mondrand werden Eintritte nur für Sterne heller als 4^m5 und Austritte für Sterne heller als 3^m5 vorhergesagt. Bei Austritten am dunklen Mondrand gilt eine Grenzgröße von 6^m5 .

Zu erwähnen ist noch, daß die Ausgabe nicht notwendig in der chronologischen Reihenfolge der einzelnen Bedeckungen erfolgt, sondern von der Reihenfolge der Einträge in der Sterndatei abhängt.

```
-----  
// Datei: Occult.cpp  
// Zweck: Berechnung von Sternbedeckungen  
// (c) 1999 Oliver Montenbruck, Thomas Pfleger  
-----
```

```
#include <cmath>  
#include <fstream>  
#include <iomanip>  
#include <iostream>  
  
#include "APC_Cheb.h"  
#include "APC_Const.h"  
#include "APC_IO.h"  
#include "APC_Math.h"  
#include "APC_Moon.h"  
#include "APC_Phys.h"
```

```

#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"

using namespace std;

// Konstanten
const double Interval = 1.0/36525.0; // 1d in Jahrhunderten

// Globale Variablen und Objekte
Cheb3D ChebMoonEqu(MoonEqu, 10, Interval);
double RA_Star;

// Klasse Star
class Star {
    ...
};

// Klasse Event
enum enEvent {dummy=-1, in=0, out=1}; // Identifiziert Eintritt oder Austritt
class Event {
public:
    Event();
    Event( const Star& aStar, double MjdUT, enEvent InOut, bool visible,
           double PosAng, double a, double b );
    inline bool IsVisible() { return m_visible; };
    friend ostream& operator << (ostream& os, Event& anEvent);
private:
    bool      m_visible;
    Star      m_Star;
    double    m_MjdUT;
    enEvent   m_InOut;
    double    m_PosAng;
    double    m_a,m_b;
};

// Konstruktoren
Event::Event() : m_visible(false), m_InOut(dummy)
{ }
Event::Event( const Star& aStar, double MjdUT, enEvent InOut, bool visible,
             double PosAng, double a, double b )
: m_visible(visible), m_Star(aStar), m_MjdUT(MjdUT), m_InOut(InOut),
  m_PosAng(PosAng), m_a(a), m_b(b)
{ }

// Ausgabeoperator
ostream& operator << (ostream& os, Event& anEvent)
{
    if ( anEvent.m_InOut != dummy ) {
        // Nur echte Ereignisse ausgeben
        os << " " << DateTime(anEvent.m_MjdUT)
        << " " << anEvent.m_Star.Name()
        << fixed << setprecision(1)
}

```

```

<< setw(5) << anEvent.m_Star.Mag()
<< "    " << ( (anEvent.m_InOut==in)? "E" : "A" )
<< "    " << Time(24.0*Modulo(anEvent.m_MjdUT, 1.0),HHMMSS)
<< fixed << setprecision(1)
<< setw(7) << int(Deg*anEvent.m_PosAng + 0.5)
<< setw(8) << anEvent.m_a/Deg
<< setw(7) << anEvent.m_b/Deg
<< endl;
}

return os;
}

...

//-----
// GetInput: Abfrage der Eingabedaten
// MjdStart Anfangszeitpunkt zur Suche nach Bedeckungen
// MjdEnd   Endzeitpunkt zur Suche nach Bedeckungen
// ET_UT    Differenz Ephemeridenzeit - Weltzeit in [s]
// R_Obs    Geozentrische Position des Beobachters in [km]
//-----

void GetInput ( double& MjdStart, double& MjdEnd,  double& ET_UT,
                Vec3D& R_Obs )
{
    int      year,month,day;
    bool     valid;
    double   Lambda, Phi;
    // Berechnungsintervall abfragen
    cout << " Zeitraum fuer die Suche nach Sternbedeckungen" << endl;
    cout << " Anfangsdatum (JJJJ MM TT)           ... ";
    cin >> year >> month >> day; cin.ignore(81,'\'n');
    MjdStart = Mjd(year,month,day);
    cout << " Enddatum       (JJJJ MM TT)           ... ";
    cin >> year >> month >> day; cin.ignore(81,'\'n');
    MjdEnd = Mjd(year,month,day);
    // Differenz Ephemeridenzeit - Weltzeit abfragen
    ETminUT ( (0.5*(MjdStart+MjdEnd)-MJD_J2000)/36525.0, ET_UT, valid );
    if ( valid ) {
        cout << " Differenz ET-UT (Vorschlag:" << fixed << setw(6)
            << setprecision (1) << ET_UT << " sec)           ... ";
        cin >> ET_UT; cin.ignore(81,'\'n');
    }
    else {
        cout << " Differenz ET-UT (sec)" << setw(33) << right << "... ";
        cin >> ET_UT; cin.ignore(81,'\'n');
    }
    // Geographische Koordinaten abfragen
    cout << " Stationskoordinaten: Laenge (oestl. pos.) [Grad] ... ";
    cin >> Lambda; cin.ignore(81,'\'n');
    cout << "                                Breite [Grad]           ... ";
    cin >> Phi; cin.ignore(81,'\'n');
    Lambda *= Rad; Phi *= Rad;
    R_Obs = Site (Lambda, Phi);
}

```

```

...
//-----
// FG: Berechnet die relativen Koordinaten von Schattenachse und Beobachter
//      im System der Fundamentalebene
//      T_ET      Ephemeridenzeit in Julianischen Jahrhunderten seit J2000
//      ET_UT     Differenz Ephemeridenzeit - Weltzeit in [s]
//      e         Richtungsvektor zum Stern (aequatoriale Koordinaten)
//      R_Obs    Geozentrische Position des Beobachters in [km]
//      f, g     Relative Koordinaten im System der Fundamentalebene
//      s         Quadrat des Abstands von Schattenachse und Beobachter
//-----
void FG ( double T_ET, double ET_UT, const Vec3D& e, const Vec3D& R_Obs,
          double& f, double& g, double& s )
{
    double MjdUT;
    Vec3D e_x, e_y, r_Moon, r_Obs;
    // Basisvektoren der Fundamentalebene
    e_x = Cross ( Vec3D(0,0,1), e );
    e_x = e_x / Norm(e_x);
    e_y = Cross ( e, e_x );
    // Mondposition
    r_Moon = ChebMoonEqu.Value(T_ET);
    // Geozentrische Position des Beobachters
    MjdUT = (36525.0*T_ET+MJD_J2000) - ET_UT/86400;
    r_Obs = R_z(-GMST(MjdUT)) * R_Obs;
    // Relative Koordinaten im System der Fundamentalebene
    f = Dot ( e_x, r_Moon - r_Obs );
    g = Dot ( e_y, r_Moon - r_Obs );
    s = f*f + g*g - R_Moon*R_Moon;
}

//-----
// Contact: Berechnet Positionswinkel, Stationskoeffizienten und prueft die
//           Sichtbarkeitsmoeglichkeiten einer Bedeckung unter Beruecksichtigung
//           der Hoehe ueber dem Horizont und der Himmelshelligkeit
//      T_ET      Ephemeridenzeit in Julianischen Jahrhunderten seit J2000
//      ET_UT     Differenz Ephemeridenzeit - Weltzeit in [s]
//      e         Richtungsvektor zum Stern (aequatoriale Koordinaten)
//      R_Obs    Geozentrische Position des Beobachters in [km]
//      InOut    zeigt Eintritt oder Austritt an
//      mag       Visuelle Helligkeit des Sterns in [mag]
//      PosAngle Positionsinkel bezogen auf Nord in [rad]
//      a         Stationskoeffizient in Laenge in [min/']
//      b         Stationskoeffizient in Breite in [min/']
//      Visible   zeigt an, ob die Bedeckung zu beobachten sein wird
//-----
void Contact ( double T_ET, double ET_UT, const Vec3D& e, const Vec3D& R_Obs,
               enEvent InOut, double mag,
               double& PosAngle, double& a, double& b, bool& Visible )
{
    const double cy = 36525.0*1440.0; // Minuten je Jahrhundert
    const double dt = 1.0;           // Schrittweite in [min]
    double MjdUT;
    Vec3D e_x, e_y, r_Moon, r_Obs, e_Obs, r_Sun;
}

```

```

Vec3D dt_dr, dr_dlambda, dr_dphi;
double s_0, f, g, ff, gg, fdot, gdot;
double Elev, ElevSun, PosAngSun, DPosAng, i;
bool BrightLimb;
// Basisvektoren der Fundamentalebene
e_x = Cross ( Vec3D(0,0,1), e );
e_x = e_x / Norm(e_x);
e_y = Cross ( e, e_x );
// Mondposition
r_Moon = ChebMoonEqu.Value(T_ET);
// Geozentrische Position des Beobachters
MjdUT = (36525.0*T_ET+MJD_J2000) - ET_UT/86400;
r_Obs = R_z(-GMST(MjdUT)) * R_Obs;
e_Obs = r_Obs / Norm(r_Obs);
// Hoehe des Sterns ueber dem Horizont
Elev = asin ( Dot(e_Obs,e) );
// Relative Koordinaten im System der Fundamentalebene
f = Dot ( e_x, r_Moon - r_Obs );
g = Dot ( e_y, r_Moon - r_Obs );
// Positionswinkel
PosAngle = Modulo(atan2(-f, -g), pi2);
// Ableitungen der Relativkoordinaten nach der Zeit
FG ( T_ET+dt/cy, ET_UT, e, R_Obs, ff, gg, s_0 );
fdot = (ff-f)/dt; // [km/min]
gdot = (gg-g)/dt;
// Ableitungen der Position des Beobachters nach geozentrischen Koordinaten
dr_dlambda = Cross ( Vec3D(0,0,1), r_Obs );
dr_dphi = Cross ( r_Obs, dr_dlambda / Norm(dr_dlambda) );
// Stationskoeffizienten
dt_dr = (f*e_x+g*e_y) / (f*fdot+g*gdot);
a = Dot ( dt_dr, dr_dlambda );
b = Dot ( dt_dr, dr_dphi );
// Berechnet Sonnenkoordinaten, Hoehe und Positionswinkel der Sonne
r_Sun = AU*SunEqu(T_ET);
ElevSun = asin ( Dot ( e_Obs, r_Sun/Norm(r_Sun) ) );
PosAngSun = PosAng(r_Moon-r_Obs, r_Sun-r_Moon);
DPosAng = Modulo(PosAngle-PosAngSun+pi, pi2) - pi;
BrightLimb = ( fabs(DPosAng) < pi/2.0 );
// Mondphase
i = acos ( Dot(-r_Moon, r_Sun-r_Moon) / (Norm(r_Sun-r_Moon)*Norm(r_Moon)) );
// Sichtbarkeitsbedingungen
Visible = ( // Mindesthoehe des Sterns
    ( Elev > +5.0*Rad ) || // 
    ( Elev > +2.0*Rad ) && ( mag <= 1.9 )
) &&
    ( // Minimale Sonnentiefe
        ( ElevSun < -6.0*Rad ) && ( mag <= 7.5 ) ||
        ( ElevSun < -3.0*Rad ) && ( mag <= 5.5 ) ||
        ( ElevSun < -0.5*Rad ) && ( mag <= 4.5 ) ||
        ( mag <= 1.9 )
) &&
    ( // Mondphase
        ( i > 36.0*Rad ) || // 
        ( i > 24.0*Rad ) && ( mag <= 6.5 ) ||
        ( i > 12.0*Rad ) && ( mag <= 5.5 ) ||

```

```

        ( mag <= 3.5 )
    ) &&
    ( // Beleuchtung am Mondrand
        ( !BrightLimb ) && (InOut==in) ||           //
        ( !BrightLimb ) && (InOut==out) && (mag<=6.5) ||
        ( BrightLimb ) && (InOut==in) && (mag<=4.5) ||
        ( BrightLimb ) && (InOut==out) && (mag<=3.5)
    );
}

//-----
// Examine: untersucht, ob eine Bedeckung eines Sterns stattfindet
// und berechnet gegebenenfalls die Daten
// T1      Anfang des Suchintervalls
// T2      Ende des Suchintervalls
// RA1     Rektaszension des Mondes zur Zeit T1 in [rad]
// RA2     Rektaszension des Mondes zur Zeit T2 in [rad]
// ET_UT   Differenz Ephemeridenzeit - Weltzeit in [s]
// R_Obs   Geozentrische Position des Beobachters in [km]
// aStar   betrachteter Stern
// In      Ereignis "Eintritt"
// Out     Ereignis "Austritt"
//-----

void Examine ( double T1, double T2, double RA1, double ET_UT,
               const Vec3D& R_Obs, Star& aStar, Event& In, Event& Out )
{
    const double cy = 876600.0;      // Stunden je Jahrhundert
    const double dT = 0.25/cy;       // Schrittweite
    const double DT = 2.25/cy;       // Suchintervall [-DT-dT,+DT+dT]
    Vec3D e;
    bool Conj, Visible;
    int n_found, n_root, i;
    double T_Conj, T, T_Cont[2], MjdUT;
    double s_minus, s_0, s_plus, xe, ye, root[2];
    double f, g, PosAng, a, b;

    In = Out = Event(); // Vorgabewerte
    // Scheinbare Koordinaten des Sterns
    e = aStar.Apparent();
    // Suche nach dem Konjunktionszeitpunkt
    Conjunction ( T1, T2, RA1, RA2, e, Conj, T_Conj );
    if (!Conj) return;
    // Suche nach den Kontaktzeiten
    n_found = 0;
    T = T_Conj - DT;
    FG (T-dT, ET_UT, e, R_Obs, f, g, s_minus);
    while (true) {
        // Interpoliere Abstand Schatten-Beobachter auf der Fundamentalebene
        // und ermittle die Kontaktzeiten
        FG (T , ET_UT, e, R_Obs, f, g, s_0 );
        FG (T+dT, ET_UT, e, R_Obs, f, g, s_plus);
        Quad (s_minus, s_0, s_plus, xe, ye, root[0], root[1], n_root);
        for (i=0;i<n_root;i++)
            T_Cont[n_found+i] = T + root[i]*dT;
        n_found += n_root;
    }
}

```

```

Conj = (n_found==2);
if ( (Conj) || (T_Conj+DT<T) ) break; // Exit loop
T+=2.0*dT; // naechster Suchzeitpunkt
s_minus = s_plus;
};

// Kontakte (Eintritt, Austritt)
if (Conj) {
    for (i=in;i<=out;i++) {
        MjdUT = 36525.0*T_Cont[i] + MJD_J2000 - ET_UT/86400.0;
        Contact ( T_Cont[i], ET_UT, e, R_Obs, enEvent(i), aStar.Mag(),
                  PosAng, a, b, Visible );
        if (i==in) In = Event(aStar, MjdUT, in , Visible, PosAng, a, b);
        if (i==out) Out = Event(aStar, MjdUT, out, Visible, PosAng, a, b);
    }
}
}

//-----
// 
// Hauptprogramm
// 
//-----
void main(int argc, char* argv[])
{
    // Konstanten
    const double Margin     = 3.0 / (36525.0*24.0); // 3h in Jahrhunderten

    // Variablen
    Star*      Stars = NULL;      // Der Sternkatalog
    int        NStars, iStar;
    double     MjdStart, MjdEnd, ET_UT, T, TEnd;
    double     RA_min, RA_max;
    Vec3D     R_Obs;
    Event     In, Out;
    char      InputFile[APC_MaxFilename] = "";
    char      OutputFile[APC_MaxFilename] = "";
    bool      FoundInputfile = false;
    bool      FoundOutputfile = false;
    ofstream  OutFile;

    // Titel
    cout << endl
        << "          OCCULT: Sternbedeckungen durch den Mond  " << endl
        << "          (c) 1999 Oliver Montenbruck, Thomas Pfleger" << endl
        << endl;
    // Ermittle die Katalogdatei und den (optionalen) Namen der Ausgabedatei
    GetFilenames( argc, argv, "Occult.dat", InputFile, FoundInputfile,
                  OutputFile, FoundOutputfile );
    // Abbruch, falls die Eingabedatei nicht gefunden wurde
    if (!FoundInputfile) { cerr << " Abbruch." << endl; exit(-1); }

    ReadCatalogue (InputFile, Stars, NStars);
    // Abbruch, falls der Sternkatalog nicht eingelesen werden konnte
    if (NStars == 0) {cerr << "Abbruch."; exit(-1); }
}

```

```

// Eingaben des Benutzers (Suchintervall, Ort u.s.w.)
GetInput (MjdStart, MjdEnd, ET_UT, R_Obs);

// Bei Bedarf die Ausgabe in eine Datei umlenken
if (FoundOutputfile) {
    OutFile.open(OutputFile);
    if (OutFile.is_open()) cout=OutFile;
}
// Vorspann ausgeben
cout
    << endl
    << "      Datum      Name      m_v     E/A      UT      Pos      a      b      "
    << endl
    << "                           mag                         Grad   m/Grad m/Grad"
    << endl;

// Suche in aufeinanderfolgenden Zeitintervallen nach Bedeckungen
// (Ephemeridenzeit)
T     = (MjdStart-MJD_J2000)/36525.0;
TEnd = (MjdEnd -MJD_J2000)/36525.0;
do {
    // Mondkoordinaten nach Tschebyscheff-Polynomen entwickeln
    ChebMoonEqu.Fit(T-Margin, T+Interval+Margin);
    RA_min = ChebMoonEqu.Value(T)[phi];
    RA_max = ChebMoonEqu.Value(T+Interval)[phi];
    // Epoche zur Berechnung der scheinbaren Sterne setzen
    Star::SetEpoch(T+Interval/2.0);
    // Schleife ueber die Katalogsterne
    for (iStar=0; iStar<NStars; iStar++) {
        // Moegliche Bedeckung untersuchen
        Examine ( T, T+Interval, RA_min, RA_max,
                  ET_UT, R_Obs, Stars[iStar], In, Out );
        // Ausgabe des Ein- und Austrittsereignisses, falls es sichtbar ist
        if (In.IsVisible() ) cout << In;
        if (Out.IsVisible()) cout << Out;
    }
    T += Interval; // Naechstes Suchintervall
}
while (T<TEnd);

if (OutFile.is_open()) OutFile.close();
if (Stars!=NULL) delete[] Stars;
}

```

Wir wollen nun als Beispiel zwei Plejadenbedeckungen aus dem Jahr 1989 mit Occult berechnen. Als Suchzeitraum wählen wir dabei die Zeit vom 15. September bis zum 15. November. Die Vorhersage soll für München gelten, das auf 11°6 östlicher Länge und 48°1 nördlicher Breite liegt. Für die Differenz zwischen Ephemeridenzeit und Weltzeit verwenden wir den von Occult aufgrund der Polynomapproximation aus Tabelle 9.2 vorgeschlagenen Wert von 57 Sekunden. Die Benutzereingaben sind durch kursive Schrift gekennzeichnet.

OCCULT: Sternbedeckungen durch den Mond
 (c) 1999 Oliver Montenbruck, Thomas Pfleger

Standard-Eingabedatei Occult.dat
 9 Sterne aus dem Katalog eingelesen

Zeitraum fuer die Suche nach Sternbedeckungen

Anfangsdatum (JJJJ MM TT)	...	1989 09 15
Enddatum (JJJJ MM TT)	...	1989 11 15
Differenz ET-UT (Vorschlag: 57.0 sec)	...	57.0
Stationskoordinaten: Laenge (oestl. pos.) [Grad]	...	11.60
Breite [Grad]	...	48.10

Datum	Name	m_v	E/A	UT	Pos	a	b
						Grad	m/Grad
1989/09/19	Celaeno	5.4	A	22:41:00	222	0.0	2.1
1989/09/19	Taygeta	4.4	E	22:01:57	67	0.1	1.7
1989/09/19	Taygeta	4.4	A	23:00:37	251	0.5	1.7
1989/09/19	Maia	4.0	E	22:15:34	98	0.5	1.3
1989/09/19	Maia	4.0	A	23:07:32	220	0.1	2.2
1989/09/19	Asterope	5.9	A	23:20:36	256	0.6	1.6
1989/11/13	Alcyone	3.0	E	19:00:10	122	1.1	0.6
1989/11/13	Alcyone	3.0	A	19:36:07	198	-0.3	2.9

Insgesamt finden in den beiden Monaten fünf Bedeckungen von Plejadensternen statt, bei denen der Ein- und/oder Austritt im Rahmen der verwendeten Kriterien gut beobachtet werden kann.

Um die Verarbeitungszeit und die Ausgabe klein zu halten, enthält die Eingabedatei des obigen Beispiels bewußt nur eine geringe Zahl von Sternen. Die beiliegende CD stellt deshalb zusätzlich eine Datei ZC.dat zur Verfügung, die 3546 Sterne des Zodiakalkatalogs für allgemeine Bedeckungsvorhersagen umfaßt. Die Positionen sind dabei mit Hilfe des PPM-Katalogs aktualisiert, der eine größere Positionsgenauigkeit bietet als der ursprüngliche Zodiakalkatalog. Zur Verwendung von ZC.dat oder einer eigenen Sterndatei gibt man beim Start von Occult nach dem Programmnamen noch die gewünschte Eingabedatei in der Kommandozeile an (vgl. Abschn. A.1.3).

10.6 Abschätzung von $\Delta T = ET - UT$ aus Beobachtungen

Hat man eine Sternbedeckung beobachtet, so ist es mit geringem Aufwand möglich, daraus einen Schätzwert für die Differenz von Weltzeit und Ephemeridenzeit zu bestimmen und damit gleichermaßen die Verzögerung der Erdrotation mitzuverfolgen.

Dazu vergleicht man den Zeitpunkt des beobachteten Ein- oder Austritts am Mondrand mit dem unter der Annahme $\Delta T = ET - UT = 0$ vorhergesagten Wert t_0 . Geht man davon aus, daß sich der Zeitpunkt t_{UT} der Beobachtung auf die Weltzeit bezieht, dann gilt zunächst für den in Ephemeridenzeit ausgedrückten Zeitpunkt die Beziehung

$$t_{ET} = t_{UT} + \Delta T .$$

Der auf die Ephemeridenzeit bezogene Zeitpunkt t_{ET} der Bedeckung ist jedoch nicht identisch mit dem Vorhersagezeitpunkt t_0 . Ursache hierfür ist, daß bei der Vorhersage mit der Annahme $ET = UT$ die Greenwichsternzeit um den Wert $\Delta\Theta_0 \approx \Delta T$ vom tatsächlichen Wert abweicht. Die berechnete Bedeckung bezieht sich damit auf einen Ort der Erdoberfläche, der um $\Delta\lambda = 15^\circ/h \cdot \Delta T$ östlich des eigentlichen Beobachtungsortes liegt. Dieser Unterschied kann jedoch auf einfache Weise mit Hilfe des Stationskoeffizienten a berücksichtigt werden. Damit gilt:

$$t_{ET} = t_0 - a (0^\circ 25/m \cdot \Delta T) .$$

Nach Einsetzen und Umformen der genannten Gleichungen erhält man hieraus die Beziehung

$$\Delta T = \frac{t_0 - t_{UT}}{1 + a \cdot 0^\circ 25/m} , \quad (10.14)$$

aus der ΔT abgeschätzt werden kann.

Tabelle 10.2. Beispiele zur Bestimmung der Differenz von Weltzeit und Ephemeridenzeit aus Sternbedeckungen

Stern	Beobachtungsort λ	Beobachtungszeit Datum	Berechnet	a	ΔT
			UT		
17 Tau	Kgl. Stw. Berlin $+13^\circ 39' +52^\circ 50'$	26.09.1896 20:32:10	20:32:02	$-0^m 7/\circ$	-10 s
57B Sco	Univ. Warschau $+21^\circ 03' +52^\circ 21'$	24.01.1930 05:30:45	05:31:08	$-0^m 3/\circ$	25 s
μ Cet	Stw. Stuttgart $+9^\circ 19' +48^\circ 78'$	10.02.1962 21:12:45	21:13:22	$+0^m 6/\circ$	32 s
δ Psc	Stw. Stuttgart $+9^\circ 19' +48^\circ 78'$	13.01.1970 19:35:12	19:36:03	$+1^m 1/\circ$	40 s
ε Ari	St. Augustin $+7^\circ 17' +50^\circ 77'$	30.11.1990 22:01:32	22:02:59	$+1^m 9/\circ$	59 s

Als Beispiel sind in Tabelle 10.2 fünf Sternbedeckungen aus den Jahren 1896 bis 1990 aufgeführt, die an verschiedenen Orten Europas beobachtet wurden. Man erkennt dabei deutlich, daß die Differenz zwischen Ephemeridenzeit und Weltzeit im Verlauf der letzten hundert Jahre um rund eine Minute angewachsen ist, was ein sichtbares Zeichen für die Abbremsung der Erdrotation darstellt. Verglichen mit den tatsächlichen Werten zeigen die aus den Beobachtungen abgeleiteten Werte für ΔT Unterschiede von 1-5 Sekunden. Diese Fehler entsprechen in etwa der von Occult erreichten Vorhersagegenauigkeit und sind im wesentlichen auf kleine Ungenauigkeiten in den Positionen der Sterne und des Mondes sowie auf die Vernachlässigung des Mondrandprofils zurückzuführen.

11. Bahnbestimmung

Die klassische Aufgabe der Bahnbestimmung besteht darin, aus möglichst wenigen beobachteten Positionen eines Planeten, Kometen oder Asteroiden dessen Bahnelemente zu ermitteln. Es handelt sich also um die Umkehrung der Ephemeridenrechnung, bei der die Positionen aus bekannten Bahnelementen berechnet werden. Jede einzelne von der Erde aus gemachte Beobachtung liefert zu einem bestimmten Zeitpunkt zwei sphärische Koordinaten. Diese können sich wahlweise auf den Himmelsäquator (Rektaszension und Deklination) oder die Ekliptik (Länge und Breite) beziehen. Die Entfernung lässt sich dagegen nicht messen und kann deshalb bei der Bahnbestimmung nicht als bekannt vorausgesetzt werden. Um sechs Bahnelemente festlegen zu können, müssen mindestens ebensoviele unabhängige Beobachtungsgrößen und damit drei Beobachtungen zur Verfügung stehen.

Die hier vorgestellte Bahnbestimmungsmethode geht auf C. F. Gauß zurück, wurde aber von H. Bucerius in verschiedenen Punkten überarbeitet und modernisiert. Den Kern des Verfahrens bildet zum einen die Festlegung der Bahnelemente aus zwei Ortsvektoren, zum anderen die Iteration der sogenannten Dreiecksflächenverhältnisse, die die geometrische Lage der drei Beobachtungspunkte in der Bahn ebene wiederspiegeln. Ergänzt wird das Verfahren durch die zwischengeschaltete Lösung der Gauß-Lagrangeschen Gleichung, die eine zuverlässige Unterscheidung mehrfacher Lösungen des Bahnbestimmungsproblems erlaubt.

11.1 Die Festlegung der Bahn durch zwei Ortsvektoren

Die Bahnelemente werden meist deshalb zur Festlegung einer Planeten- oder Kometenbahn verwendet, weil sie eine besonders anschauliche Interpretation der einzelnen Größen erlauben. Für die Bahnbestimmung erweist sich allerdings eine andere Beschreibung als günstiger. Die Bahn ist nämlich ebenso eindeutig definiert, wenn man den Ort und die Geschwindigkeit des Himmelskörpers zu einem Zeitpunkt kennt oder wahlweise zwei Bahnpunkte und die Zeiten, zu denen sie passiert werden. Während die erste Darstellung in der Laplaceschen Bahnbestimmung verwendet wird, bedient sich die Gaußsche Methode der Beschreibung durch zwei Ortsvektoren. Die Aufgabe dieses Abschnitts wird deshalb darin bestehen, die Elemente einer Bahn zu berechnen, die durch zwei gegebene Orte \mathbf{r}_a und \mathbf{r}_b sowie die zugehörigen Zeiten t_a und t_b festgelegt ist. Damit reduziert sich das Problem der Bahnbestimmung später darauf, zwei heliozentrische Orte aus den drei beobachteten geozentrischen Richtungen zu ermitteln.

Als Zwischenschritt wird zunächst die Berechnung des *Sektor-zu-Dreieck-Verhältnisses* behandelt, die den schwierigsten Teil der Bahnelementeberechnung dar-

stellt. Daneben spielt diese Größe auch für den weiteren Gang der Bahnbestimmung eine zentrale Rolle, die allerdings erst in den folgenden Abschnitten deutlich wird.

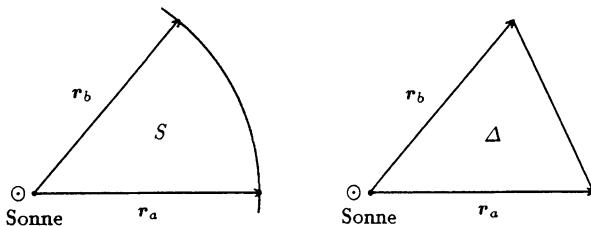


Abb. 11.1.
Sektor- und Dreiecksfläche

11.1.1 Das Sektor-zu-Dreieck-Verhältnis

Die Fläche Δ des von den Vektoren \mathbf{r}_a und \mathbf{r}_b aufgespannten Dreiecks (Abb. 11.1) hängt von den Seitenlängen r_a und r_b sowie dem Zwischenwinkel $\nu_b - \nu_a$ ab, der im folgenden immer kleiner als 180° sein soll:

$$\Delta = \frac{1}{2} r_a r_b \cdot \sin(\nu_b - \nu_a) . \quad (11.1)$$

ν_a und ν_b bezeichnen dabei die Werte der wahren Anomalie in den Randpunkten der Bahn.

Die Sektorfläche S , die von \mathbf{r}_a , \mathbf{r}_b und dem dazwischenliegenden Bahnbogen begrenzt wird, ist aufgrund des zweiten Keplerschen Gesetzes (Flächensatz) proportional zur Differenz der beiden Zeiten t_a und t_b :

$$S = \frac{1}{2} \sqrt{GM_\odot} \cdot \sqrt{a(1-e^2)} \cdot (t_b - t_a) . \quad (11.2)$$

Hierin bezeichnen a und e die Halbachse und die Exzentrizität der Bahn, die die vorgegebenen Punkte verbindet (vgl. Kap. 4). Führt man noch den Bahnpараметer $p = a(1 - e^2)$ ein, dann ergibt sich für das Verhältnis η der beiden Flächen der Ausdruck

$$\eta = \frac{S}{\Delta} = \frac{\sqrt{p} \cdot \tau}{r_a r_b \cdot \sin(\nu_b - \nu_a)} , \quad (11.3)$$

in dem zur Abkürzung für die Zwischenzeit die Größe

$$\tau = \sqrt{GM_\odot} \cdot (t_b - t_a) \quad (11.4)$$

definiert wurde. Wie man sieht, enthält die Formel für η noch den Bahnpараметer p , der bisher nicht durch die Vektoren \mathbf{r}_a und \mathbf{r}_b ausgedrückt wurde. Versucht man, den Bahnpараметer mit Hilfe der bekannten Formeln für das Zweikörperproblem zu eliminieren, dann zeigt sich allerdings, daß es nicht mehr möglich ist, η durch einen geschlossenen Ausdruck darzustellen. Man findet stattdessen die transzendentale Gleichung¹

$$\eta = 1 + \frac{m}{\eta^2} \cdot W \left(\frac{m}{\eta^2} - l \right) , \quad (11.5)$$

¹Auf eine Herleitung dieser Gleichung, die sich über mehrere Seiten erstreckt, wurde hier verzichtet. Der Leser sei dazu auf die im Anhang angeführte Literatur verwiesen.

mit den (positiven) Hilfsgrößen

$$\begin{aligned} m &= \frac{\tau^2}{\sqrt{2(r_a r_b + \mathbf{r}_a \cdot \mathbf{r}_b)}^3} \\ l &= \frac{r_a + r_b}{2\sqrt{2(r_a r_b + \mathbf{r}_a \cdot \mathbf{r}_b)}} - \frac{1}{2} \quad , \end{aligned} \quad (11.6)$$

aus der η iterativ bestimmt werden muß. Die Funktion W ist dabei abschnittsweise wie folgt definiert:

$$W(w) = \begin{cases} \frac{2g - \sin(2g)}{\sin^3(g)}, & g = 2 \arcsin \sqrt{w} \quad 0 < w < 1 \\ \frac{4}{3} + \frac{4 \cdot 6}{3 \cdot 5} w + \frac{4 \cdot 6 \cdot 8}{3 \cdot 5 \cdot 7} w^2 + \dots & w \approx 0 \\ \frac{\sinh(2g) - 2g}{\sinh^3(g)}, & g = 2 \operatorname{arsinh} \sqrt{-w} \quad w < 0 \end{cases} . \quad (11.7)$$

Schreibt man

$$f(x) = 1 - x + \frac{m}{x^2} \cdot W\left(\frac{m}{x^2} - l\right) \quad ,$$

dann ist der gesuchte Wert von η gerade die Nullstelle der Funktion f .

```
// F : Lokale, von FindEta() benutzte Funktion
// F = 1 - eta + (m/eta**2)*W(m/eta**2-l)
double F (double eta, double m, double l)
{
    const double eps = 100.0 * eps_mach;
    double w,W,a,n,g;
    w = m/(eta*eta)-l;
    if (fabs(w)<0.1) { // Reihenentwicklung
        W = a = 4.0/3.0; n = 0.0;
        do { n+=1.0; a*=w*(n+2.0)/(n+1.5); W+=a; } while (fabs(a)>=eps);
    }
    else {
        if (w > 0.0) {
            g = 2.0*asin(sqrt(w));
            W = (2.0*g - sin(2.0*g)) / pow(sin(g), 3);
        }
        else {
            g = 2.0*log(sqrt(-w)+sqrt(1.0-w)); // =2.0*arsinh(sqrt(-w))
            W = (sinh(2.0*g) - 2.0*g) / pow(sinh(g), 3);
        }
    }
    return ( 1.0 - eta + (w+l)*W );
}
```

Da die Standardbibliothek der Sprache C++ keine inversen Hyperbelfunktionen kennt, wurde die Funktion $\operatorname{arsinh} x = \ln(x + \sqrt{1 + x^2})$ mit Hilfe des natürlichen Logarithmus ersetzt.

Zur iterativen Berechnung von η eignet sich das Sekantenverfahren. Ausgehend von zwei Näherungswerten η_{i-1} und η_i erhält man über

$$\eta_{i+1} = \eta_i - f(\eta_i) \cdot \frac{\eta_i - \eta_{i-1}}{f(\eta_i) - f(\eta_{i-1})}$$

einen verbesserten Wert η_{i+1} . Anschaulich gesprochen wird dabei die Nullstelle der Sekante bestimmt, die durch die Punkte $(\eta_{i-1}, f(\eta_{i-1}))$ und $(\eta_i, f(\eta_i))$ des Graphen von f verläuft. Führt man diesen Schritt mehrmals hintereinander durch, dann liefert die Iteration rasch den gesuchten Wert für das Sektor- zu Dreiecksverhältnis. Geeignete Startwerte

$$\eta_1 = \eta_{\text{Hansen}} + 0.1 \quad \text{und} \quad \eta_2 = \eta_{\text{Hansen}}$$

ergeben sich aus der sogenannten Hansenschen Näherung

$$\eta_{\text{Hansen}} = \frac{12}{22} + \frac{10}{22} \sqrt{1 + \frac{44}{9} \frac{m}{l + 5/6}} \quad . \quad (11.8)$$

Damit erhält man die folgende Funktion zur Berechnung des Sektor-zu-Dreieck-Verhältnisses:

```
-----
// FindEta: Berechnet das Sektor-zu-Dreiecks-Verhaeltnis aus zwei gegebenen
// Orten und der dazugehoerigen Zwischenzeit
// r_a          Ort zum Zeitpunkt A in [AE]
// r_b          Ort zum Zeitpunkt B in [AE]
// tau         Zeit zwischen den Passagen von A und B (kGauss * dT in [d])
// <return>    Sektor-zu-Dreiecks-Verhaeltnis
-----

double FindEta (const Vec3D& r_a, const Vec3D& r_b, double tau)
{
    const int      maxit = 30;
    const double   delta = 100.0*eps_mach;
    int           i;
    double kappa, m, l, s_a, s_b, eta_min, eta1, eta2, F1, F2, d_eta;
    s_a = Norm(r_a); s_b = Norm(r_b);
    kappa = sqrt ( 2.0*(s_a*s_b+Dot(r_a,r_b)) );
    m = tau*tau/pow(kappa,3); l = (s_a+s_b)/(2.0*kappa)-0.5;
    eta_min = sqrt(m/(l+1.0));
    // Start mit der Hansen'schen Naehlerung
    eta2 = ( 12.0 + 10.0*sqrt(1.0+(44.0/9.0)*m /(l+5.0/6.0)) ) / 22.0;
    eta1 = eta2 + 0.1;
    // Sekantenverfahren
    F1 = F(eta1,m,l); F2 = F(eta2,m,l); i = 0;
    while (fabs(F2-F1) > delta)
    {
        d_eta = -F2*(eta2-eta1)/(F2-F1);
        eta1=eta2; F1=F2; while (eta2+d_eta<=eta_min) d_eta*=0.5;
        eta2+=d_eta; F2=F(eta2,m,l); ++i;
        if (i==maxit) { cerr << "Konvergenzprobleme in FindEta" << endl; break; }
    }
    return eta2;
}
```

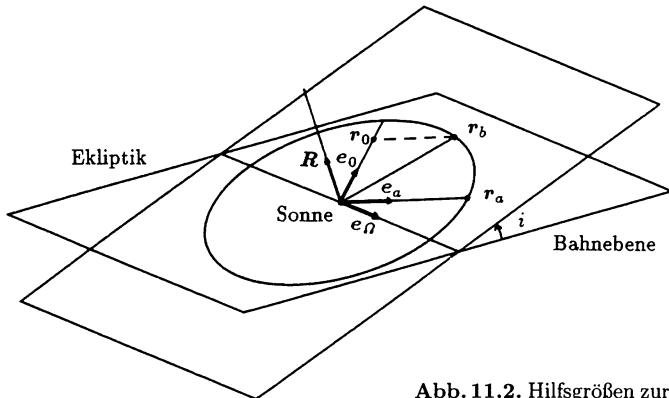


Abb. 11.2. Hilfsgrößen zur Festlegung der Bahnebene

11.1.2 Die Bahnelemente

Die Bahn eines Himmelskörpers, die durch die Punkte \mathbf{r}_a und \mathbf{r}_b verläuft, bleibt ständig auf die Ebene beschränkt, die durch diese beiden Punkte und die Sonne festgelegt wird. Um die Neigung i dieser Ebene gegen die Ekliptik sowie die Knotenlänge Ω zu bestimmen, konstruiert man zunächst die Einheitsvektoren \mathbf{e}_a und \mathbf{e}_0 , die zusammen die Bahnebene aufspannen:

$$\mathbf{e}_a = \frac{\mathbf{r}_a}{|\mathbf{r}_a|} \quad (11.9)$$

$$\mathbf{e}_0 = \frac{\mathbf{r}_0}{|\mathbf{r}_0|} \quad \text{mit} \quad \mathbf{r}_0 = \mathbf{r}_b - (\mathbf{r}_b \cdot \mathbf{e}_a)\mathbf{e}_a \quad . \quad (11.10)$$

Die Bedeutung dieser Vektoren ist aus Abb. 11.2 ersichtlich. Während \mathbf{e}_a in Richtung von \mathbf{r}_a weist, zeigen \mathbf{r}_0 und \mathbf{e}_0 in die dazu senkrechte Richtung. Bildet man nun das Kreuzprodukt von \mathbf{e}_a und \mathbf{e}_0 , so erhält man als Resultat den Gaußschen Vektor \mathbf{R} , der auf der Bahnebene senkrecht steht und ebenfalls auf die Länge Eins normiert ist ($|\mathbf{R}| = 1$):

$$\mathbf{R} = \mathbf{e}_a \times \mathbf{e}_0 \quad , \quad \begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix} = \begin{pmatrix} y_a z_0 - z_a y_0 \\ z_a x_0 - x_a z_0 \\ x_a y_0 - y_a x_0 \end{pmatrix} \quad . \quad (11.11)$$

\mathbf{R} zeigt in Richtung der ekliptikalnen Länge $l = \Omega - 90^\circ$ und der Breite $b = 90^\circ - i$ und kann deshalb auch durch die Lageelemente Ω und i ausgedrückt werden:

$$\mathbf{R} = \begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix} = \begin{pmatrix} +\cos(90^\circ - i) \cos(\Omega - 90^\circ) \\ +\cos(90^\circ - i) \sin(\Omega - 90^\circ) \\ +\sin(90^\circ - i) \end{pmatrix} = \begin{pmatrix} +\sin i \sin \Omega \\ -\sin i \cos \Omega \\ +\cos i \end{pmatrix} \quad . \quad (11.12)$$

Auf diese Weise erhält man drei Gleichungen, die sich in eindeutiger Weise nach der Knotenlänge und der Bahnneigung auflösen lassen:

$$\Omega = 90^\circ + \arctan(R_y/R_x) = \arctan(-R_x/R_y) \quad (11.13)$$

$$i = 90^\circ - \arcsin(R_z) \quad . \quad (11.14)$$

Beide Winkel beziehen sich auf das gleiche Äquinoktium wie die Vektoren \mathbf{r}_a und \mathbf{r}_b . Aus der Lage der Knotenlinie lässt sich nun das Argument der Breite u_a berechnen, das den Winkel zwischen dem Ortsvektor \mathbf{r}_a und der Richtung zum aufsteigenden Knoten der Bahn angibt. Für diesen Winkel gilt

$$\begin{aligned}\cos u_a &= \mathbf{e}_a \cdot \mathbf{e}_\Omega = x_a \cdot \cos \Omega + y_a \cdot \sin \Omega \\ \cos(u_a + 90^\circ) &= \mathbf{e}_0 \cdot \mathbf{e}_\Omega = x_0 \cdot \cos \Omega + y_0 \cdot \sin \Omega \quad ,\end{aligned}$$

wobei

$$\mathbf{e}_\Omega = \begin{pmatrix} \cos \Omega \\ \sin \Omega \\ 0 \end{pmatrix}$$

den Einheitsvektor in Richtung der Knotenlinie darstellt. Damit ist

$$\begin{aligned}u_a &= \arctan \left(\frac{-x_0 \cdot \cos \Omega - y_0 \cdot \sin \Omega}{+x_a \cdot \cos \Omega + y_a \cdot \sin \Omega} \right) \\ &= \arctan \left(\frac{+x_0 \cdot R_y - y_0 \cdot R_x}{-x_a \cdot R_y + y_a \cdot R_x} \right) \quad .\end{aligned}\tag{11.15}$$

Für die Bestimmung der weiteren Bahnparameter benötigt man das Sektor- zu Dreiecksverhältnis, dessen Berechnung bereits im letzten Abschnitt vorgestellt wurde. Mit seiner Hilfe lässt sich zunächst der Bahnparameter

$$p = \left(\frac{2 \cdot \Delta \cdot \eta}{\tau} \right)^2$$

über die von \mathbf{r}_a und \mathbf{r}_b aufgespannte Dreiecksfläche

$$\Delta = \frac{1}{2} r_a r_b \cdot \sin(\nu_b - \nu_a) = \frac{1}{2} r_a r_0$$

und die Zwischenzeit τ darstellen.

Die Form der Bahn wird durch die Exzentrizität e beschrieben, die sich über die Kegelschnittsgleichung

$$r = \frac{p}{1 + e \cdot \cos \nu}$$

bestimmen lässt. Aufgelöst nach $e \cos \nu$ gilt für die gegebenen Bahnpunkte der Zusammenhang

$$\begin{aligned}e \cdot \cos \nu_a &= p/r_a - 1 \\ e \cdot \cos \nu_b &= p/r_b - 1 \quad .\end{aligned}$$

Berücksichtigt man noch die Identität

$$\begin{aligned}\cos \nu_b &= \cos \nu_a \cos(\nu_b - \nu_a) - \sin \nu_a \sin(\nu_b - \nu_a) \\ &= \cos \nu_a \cdot \left(\frac{\mathbf{r}_b \cdot \mathbf{e}_a}{r_b} \right) - \sin \nu_a \cdot \left(\frac{r_0}{r_b} \right) \quad ,\end{aligned}$$

dann erhält man zusammengefaßt die zwei Gleichungen

$$\begin{aligned} e \cdot \cos \nu_a &= p/r_a - 1 \\ e \cdot \sin \nu_a &= \left\{ (p/r_a - 1) \left(\frac{\mathbf{r}_b \cdot \mathbf{e}_a}{r_b} \right) - (p/r_b - 1) \right\} / \left(\frac{r_0}{r_b} \right) \quad . \end{aligned}$$

die ihrerseits nach der Exzentrizität und der wahren Anomalie zur Zeit t_a aufgelöst werden können:

$$\begin{aligned} e &= \sqrt{(e \cdot \cos(\nu_a))^2 + (e \cdot \sin(\nu_a))^2} \\ \nu_a &= \arctan \left(\frac{e \cdot \sin(\nu_a)}{e \cdot \cos(\nu_a)} \right) \quad . \end{aligned}$$

Aus der Differenz des Arguments der Breite und der wahren Anomalie ergeben sich das Argument und die Länge des Perihels zu

$$\omega = u_a - \nu_a \quad (11.16)$$

$$\varpi = u_a - \nu_a + \Omega \quad . \quad (11.17)$$

Anhand der Exzentrizität lässt sich feststellen, ob es sich bei der Bahn, die \mathbf{r}_a und \mathbf{r}_b verbindet, um eine Ellipse ($e < 1$) oder Hyperbel ($e > 1$) handelt. Parabelbahnen werden hier nicht weiter berücksichtigt, weil es in der Praxis so gut wie ausgeschlossen ist, daß e bei einer Bahnbestimmung exakt gleich Eins wird. Aus dem Bahnpараметer und der Exzentrizität erhält man nun auch die große Halbachse der Bahn und die Periheldistanz:

$$a = \frac{p}{1 - e^2} \quad (11.18)$$

$$q = \frac{p}{1 + e} \quad . \quad (11.19)$$

Zu beachten ist dabei, daß die große Halbachse einer Hyperbelbahn definitionsgemäß eine negative Größe ist.

Damit sind nun alle Bahnelemente bekannt, die die Form (e), die Größe (a) und die räumliche Lage (i, Ω, ω) der Bahn festlegen. Als sechstes und letztes Element bleibt somit die Perihelzeit zu ermitteln, die angibt, wann der sonnennächste Bahnpunkt passiert wird oder wurde. Dazu wird für Ellipsenbahnen zunächst die exzentrische Anomalie E_a aus den Gleichungen

$$\cos E_a = \frac{\cos \nu_a + e}{1 + e \cdot \cos \nu_a}$$

$$\sin E_a = \frac{\sqrt{1 - e^2} \sin \nu_a}{1 + e \cdot \cos \nu_a}$$

bestimmt. Man erhält diese Gleichungen, wenn man in (4.5) den Radius mit Hilfe der Kegelschnittgleichung eliminiert und anschließend nach der exzentrischen Anomalie auflöst. Der zu E gehörige Wert der mittleren Anomalie M folgt dann aus der Keplergleichung

$$M_a = E_a - e \cdot \sin E_a \quad (\text{Bogenmaß}) \quad .$$

Bei einer Umlaufszeit von

$$T = 2\pi \cdot \sqrt{\frac{a^3}{GM_{\odot}}} ,$$

wie sie sich aus dem dritten Keplerschen Gesetz ergibt, ändert sich die mittlere Anomalie täglich um

$$n = 2\pi \cdot \frac{1^d}{T} = \sqrt{\frac{GM_{\odot}}{a^3}} \cdot 1^d ,$$

so daß der Zeitpunkt des Periheldurchgangs durch

$$t_0 = t_a - M_a / \sqrt{GM_{\odot}/a^3} \quad (11.20)$$

gegeben ist. Für Hyperbelbahnen lauten die entsprechenden Beziehungen

$$\sinh H_a = \frac{\sqrt{e^2 - 1} \sin \nu_a}{1 + e \cdot \cos \nu_a} \quad (11.21)$$

$$M_a = e \cdot \sinh H_a - H_a \quad (11.22)$$

$$t_0 = t_a - M_a / \sqrt{GM_{\odot}/|a|^3} . \quad (11.23)$$

Die vollständige Bestimmung der Bahnelemente aus zwei vorgegebenen Bahnpunkten ist in der Funktion **Elements** zusammengefaßt.

```
-----
// Elements: Berechnet die Elemente einer Keplerbahn aus zwei gegebenen Orten
// GM      Produkt aus Gravitationskonstante und Zentralmasse [AE^3*d^-2]
// Mjd_a   Zeitpunkt der Passage des Ortes A [MJD]
// Mjd_b   Zeitpunkt der Passage des Ortes B [MJD]
// r_a     Ort A in heliozentrischen ekliptikalnen Koordinaten in [AU]
// r_b     Ort B in heliozentrischen ekliptikalnen Koordinaten in [AU]
// Mjd_p   Zeitpunkt des Periheldurchgangs
// q       Periheldistanz in [AE]
// i       Neigung der Bahn zur Ekliptik in [rad]
// Omega   Laenge des aufsteigenden Knotens der Bahn in [rad]
// omega   Argument des Perihels in [rad]
// M       Mittlere Anomalie in [rad]
//-
void Elements ( double GM, double Mjd_a, double Mjd_b,
                 const Vec3D& r_a, const Vec3D& r_b,
                 double& Mjd_p, double& q, double& e,
                 double& i, double& Omega, double& omega )
{
    double tau, eta, p, a, n, nu, E, M, u;
    double s_a, s_b, s_0, fac, sinhH;
    double cos_dnu, sin_dnu, ecos_nu, esin_nu;
    Vec3D e_a, r_0, e_0, R;
    // Berechne Vektor r_0 (Anteil von r_b, der senkrecht auf r_a steht)
    // und die Betraege der Vektoren r_a,r_b und r_0
    s_a = Norm(r_a); e_a = r_a/s_a;
    s_b = Norm(r_b);
    fac = Dot(r_b,e_a); r_0 = r_b-fac*e_a;
    s_0 = Norm(r_0); e_0 = r_0/s_0;
```

```

// Bahnneigung und aufsteigender Knoten
R      = Cross(e_a,e_0);
i      = pi/2.0 - R[theta];
Omega = Modulo ( pi/2.0 + R[phi], 2.0*pi );
if (i==0.0)
    u = atan2 ( r_a[y], r_a[x] );
else
    u = atan2 ( (+e_0[x]*R[y]-e_0[y]*R[x]) , (-e_a[x]*R[y]+e_a[y]*R[x]) );
// Semilatus rectum
tau = sqrt(GM) * fabs(Mjd_b-Mjd_a);
eta = FindEta ( r_a, r_b, tau );
p   = pow ( s_a*s_0*eta/tau, 2 );
// Exzentrizitaet, wahre Anomalie und Argument des Perihels
cos_dnu = fac / s_b;
sin_dnu = s_0 / s_b;
ecos_nu = p / s_a - 1.0;
esin_nu = ( ecos_nu * cos_dnu - (p/s_b-1.0) ) / sin_dnu;
e   = sqrt ( ecos_nu*ecos_nu + esin_nu*esin_nu );
nu = atan2(esin_nu,ecos_nu);
omega = Modulo(u-nu,2.0*pi);
// Periheldistanz, grosse Halbachse und mittlere Bewegung
q = p/(1.0+e);
a = q/(1.0-e);
n = sqrt ( GM / fabs(a*a*a) );
// Mittlere Anomalie und Zeitpunkt des Periheldurchgangs
if (e<1.0)
    E = atan2 ( sqrt((1.0-e)*(1.0+e)) * esin_nu,   ecos_nu + e*e );
    M = E - e*sin(E);
}
else
{
    sinhH = sqrt((e-1.0)*(e+1.0)) * esin_nu / ( e + e * ecos_nu );
    M = e * sinhH - log ( sinhH + sqrt(1.0+sinhH*sinhH) );
}
Mjd_p = Mjd_a - M / n;
}

```

11.2 Das verkürzte Gauß-Verfahren

11.2.1 Geometrie der geozentrischen Beobachtungen

Ist \mathbf{r} der heliozentrische (auf die Sonne bezogene) Ort eines Planeten und \mathbf{R} der geozentrische (auf die Erde bezogene) Ort der Sonne, dann ist der geozentrische Planetenort durch

$$\rho \mathbf{e} = \mathbf{R} + \mathbf{r} \quad (11.24)$$

gegeben. Darin ist ρ die Entfernung des Planeten von der Erde und \mathbf{e} ein Vektor der Länge Eins, der von der Erde zum Planeten zeigt. In ekliptikalnen oder äquatorialen Koordinaten hat \mathbf{e} die Komponenten

$$\begin{pmatrix} \cos \lambda \cos \beta \\ \sin \lambda \cos \beta \\ \sin \beta \end{pmatrix} \quad \text{bzw.} \quad \begin{pmatrix} \cos \alpha \cos \delta \\ \sin \alpha \cos \delta \\ \sin \delta \end{pmatrix},$$

wenn λ und β die ekliptikale Länge und Breite und α und δ die Rektaszension und Deklination des Planeten vom Standpunkt der Erde aus bezeichnen.

Mißt man die Koordinaten des Planeten an der scheinbaren Himmelskugel (etwa α und δ), dann ist dadurch nur die Beobachtungsrichtung – also e – festgelegt. Die Entfernung ρ ist unbekannt und muß im Rahmen der Bahnbestimmung berechnet werden. Für die eindeutige Bestimmung einer Planetenbahn müssen deshalb mindestens drei Beobachtungen e_1, e_2, e_3 vorliegen. Die Koordinaten R_1, R_2, R_3 der Sonne zu den Beobachtungszeitpunkten können zusätzlich als bekannt vorausgesetzt werden. Ausgehend von diesen Daten muß man versuchen, die Entfernungen ρ_1, ρ_2 und ρ_3 zu berechnen. Erst wenn diese bekannt sind, lassen sich auch die heliozentrischen Orte bestimmen, die die Bahn festlegen und eine Ableitung der Bahnelemente erlauben. Die geometrischen Beziehungen, die sich für einen Satz von drei Planetenpositionen ergeben, sollen nun in einer für die Zwecke der Bahnbestimmung geeigneten Form abgeleitet werden.

Zu den Zeiten $t_1 < t_2 < t_3$ befindet sich der Planet an den Orten r_1, r_2, r_3 relativ zur Sonne. Da im Rahmen einer ungestörten Keplerbewegung alle Orte in einer Ebene mit der Sonne liegen, ist es immer möglich, einen Ortsvektor durch eine geeignete Kombination der beiden anderen auszudrücken. Man wählt dazu r_2 und kann dann schreiben

$$r_2 = n_1 r_1 + n_3 r_3 \quad (\text{Ebenengleichung}) \quad . \quad (11.25)$$

Die Faktoren n_1 und n_3 hängen von der relativen Lage von r_1, r_2 und r_3 ab. Setzt man im folgenden voraus, daß der gesamte Bahnbogen kleiner als 180° ist, dann sind beide Faktoren positiv. Kombiniert man nun die (11.24) und (11.25), dann erhält man

$$(\rho_2 e_2 - R_2) = n_1 \cdot (\rho_1 e_1 - R_1) + n_3 \cdot (\rho_3 e_3 - R_3)$$

oder umgestellt

$$n_1 \rho_1 e_1 - \rho_2 e_2 + n_3 \rho_3 e_3 = n_1 R_1 - R_2 + n_3 R_3 \quad . \quad (11.26)$$

Definiert man nun die Vektoren

$$d_1 = e_2 \times e_3 \quad d_2 = e_3 \times e_1 \quad d_3 = e_1 \times e_2 \quad ,$$

dann steht aufgrund der Eigenschaften des Kreuzproduktes d_1 senkrecht auf e_2 und e_3 , d_2 senkrecht auf e_3 und e_1 und d_3 senkrecht auf e_1 und e_2 . Folglich ist das Skalarprodukt $e_i \cdot d_j$ nur für $i = j$ von Null verschieden. Multipliziert man (11.26) jeweils mit d_1, d_2 und d_3 , dann erhält man nacheinander die Gleichungen

$$\begin{aligned} n_1 \rho_1 \cdot (e_1 \cdot d_1) &= (n_1 R_1 - R_2 + n_3 R_3) \cdot d_1 \\ -\rho_2 \cdot (e_2 \cdot d_2) &= (n_1 R_1 - R_2 + n_3 R_3) \cdot d_2 \\ n_3 \rho_3 \cdot (e_3 \cdot d_3) &= (n_1 R_1 - R_2 + n_3 R_3) \cdot d_3 \quad . \end{aligned}$$

Führt man noch die Abkürzungen

$$\begin{aligned} D &= e_1 \cdot (e_2 \times e_3) = e_2 \cdot (e_3 \times e_1) = e_3 \cdot (e_1 \times e_2) \\ &= e_1 \cdot d_1 &= e_2 \cdot d_2 &= e_3 \cdot d_3 \end{aligned}$$

und

$$D_{ij} = \mathbf{d}_i \cdot \mathbf{R}_j$$

ein, dann ergeben sich damit die drei Gleichungen

$$\begin{aligned}\rho_1 &= \frac{1}{n_1 D}(n_1 D_{11} - D_{12} + n_3 D_{13}) \\ \rho_2 &= \frac{1}{-D}(n_1 D_{21} - D_{22} + n_3 D_{23}) \\ \rho_3 &= \frac{1}{n_3 D}(n_1 D_{31} - D_{32} + n_3 D_{33})\end{aligned}\quad . \quad (11.27)$$

Die Entfernungen ρ_1 , ρ_2 und ρ_3 können also durch n_1 und n_3 sowie die Vektoren $\mathbf{e}_{1\dots 3}$ und $\mathbf{R}_{1\dots 3}$ ausgedrückt werden. Damit scheint zunächst nur wenig gewonnenen, da n_1 und n_3 ja nicht bekannt sind. Immerhin hat sich die Zahl der Unbekannten durch Ausnutzung der Ebenengleichung von drei ($\rho_{1\dots 3}$) auf zwei ($n_{1,3}$) verringert. Wirklich bedeutsam werden die neu eingeführten Koeffizienten jedoch dadurch, daß sie sich – wie nun gezeigt wird – in guter Näherung durch Verhältnisse der bekannten Zwischenzeiten darstellen lassen.

Dazu wird noch einmal die Ebenengleichung (11.25) betrachtet. Bildet man auf beiden Seiten das Kreuzprodukt mit \mathbf{r}_3 oder \mathbf{r}_1 und beachtet, daß das Kreuzprodukt eines Vektors mit sich selbst verschwindet, dann folgen daraus die Beziehungen

$$(\mathbf{r}_2 \times \mathbf{r}_3) = n_1 \cdot (\mathbf{r}_1 \times \mathbf{r}_3) \quad (\mathbf{r}_1 \times \mathbf{r}_2) = n_3 \cdot (\mathbf{r}_1 \times \mathbf{r}_3)$$

und

$$n_1 = \frac{|\mathbf{r}_2 \times \mathbf{r}_3|}{|\mathbf{r}_1 \times \mathbf{r}_3|} \quad n_3 = \frac{|\mathbf{r}_1 \times \mathbf{r}_2|}{|\mathbf{r}_1 \times \mathbf{r}_3|} \quad .$$

Da allgemein die Dreiecksfläche Δ , die von zwei Vektoren \mathbf{r}_a und \mathbf{r}_b aufgespannt wird, gleich

$$\Delta = \frac{1}{2} |\mathbf{r}_a \times \mathbf{r}_b|$$

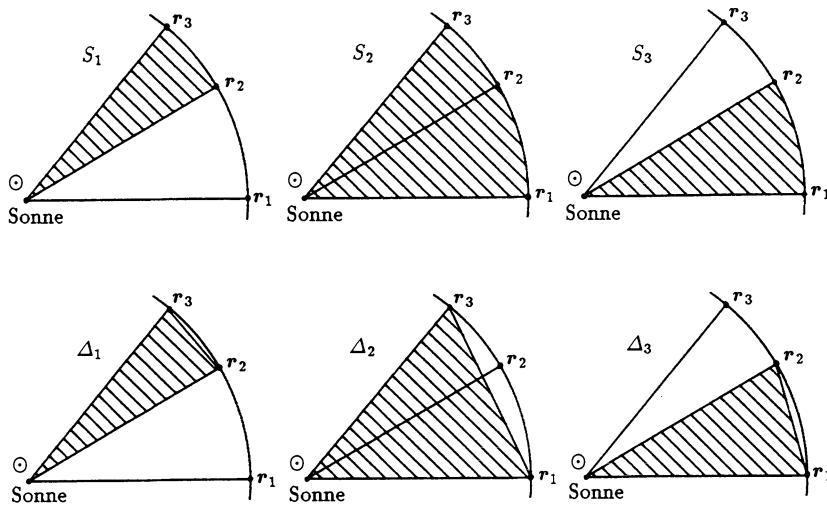
ist, lassen sich n_1 und n_3 als Verhältnisse der Dreiecksflächen interpretieren, die von \mathbf{r}_1 , \mathbf{r}_2 und \mathbf{r}_3 aufgespannt werden (vgl. Abb. 11.3):

$$n_1 = \frac{\Delta_1}{\Delta_2} \quad n_3 = \frac{\Delta_3}{\Delta_2} \quad .$$

Besonders für kleine Bahnbögen unterscheiden sich die Dreiecksflächen nur wenig von den entsprechenden Sektorflächen $S_i = \eta_i \Delta_i$, die ihrerseits proportional zu den Zwischenzeiten τ_i sind:

$$n_1 = \frac{\eta_2}{\eta_1} \cdot \frac{\tau_1}{\tau_2} \approx \frac{\tau_1}{\tau_2} \quad n_3 = \frac{\eta_2}{\eta_3} \cdot \frac{\tau_3}{\tau_2} \approx \frac{\tau_3}{\tau_2} \quad . \quad (11.28)$$

Somit sind zumindest ungefähre Werte für die n_1 und n_3 bekannt, die über (11.27) eine erste nährungsweise Bestimmung der geozentrischen Entfernungen (ρ_1, ρ_2, ρ_3) erlauben.



$$\tau_1 = \sqrt{GM_{\odot}}(t_3 - t_2)$$

$$\tau_2 = \sqrt{GM_{\odot}}(t_3 - t_1)$$

$$\tau_3 = \sqrt{GM_{\odot}}(t_2 - t_1)$$

Abb. 11.3. Sektorflächen, Dreiecksflächen und Zwischenzeiten für drei heliozentrische Orte

11.2.2 Iteration der Dreiecksflächenverhältnisse

Die bisherigen Überlegungen lassen sich im wesentlichen zu zwei Aussagen zusammenfassen. Kennt man den heliozentrischen Ort eines Himmelskörpers an zwei vorgegebenen Zeitpunkten, dann ist der gesamte Bahnverlauf dadurch eindeutig festgelegt. Gleiches gilt für die Bahnelemente und das Sektor- zu Dreiecksverhältnis. Kennt man andererseits die Werte der Sektor- zu Dreiecksverhältnisse zu einem Satz von drei beobachteten Positionen, dann lassen sich daraus die geozentrischen und heliozentrischen Ortsvektoren berechnen. Hierauf baut die nun beschriebene verkürzte Gaußsche Bahnbestimmungsmethode auf.

Gegeben seien dazu die drei Beobachtungsrichtungen (e_1, e_2, e_3), die zugehörigen geozentrischen Koordinaten der Sonne ($\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$) und die Zwischenzeiten (τ_1, τ_2, τ_3). Mit diesen Ausgangsdaten werden die folgenden Schritte durchgeführt:

1. Setze $n_1 = \tau_1/\tau_2$ und $n_3 = \tau_3/\tau_2$ als Ausgangsnäherung für die Dreiecksflächenverhältnisse.
2. Wiederhole die Schritte (a) ... (d), bis sich n_1, n_3 und die übrigen Größen nicht mehr wesentlich verändern.
 - (a) Berechne die geozentrischen Entferungen (ρ_1, ρ_2, ρ_3) nach (11.27).
 - (b) Berechne hiermit die heliozentrischen Positionen ($\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$) aus (11.24).
 - (c) Berechne die Sektor- zu Dreiecksverhältnisse (η_1, η_2, η_3) aus je zwei heliozentrischen Positionen und der zugehörigen Zwischenzeit nach (11.5).
 - (d) Berechne verbesserte Werte $n_1 = (\eta_2/\eta_1) \cdot (\tau_1/\tau_2)$ und $n_3 = (\eta_2/\eta_3) \cdot (\tau_3/\tau_2)$ für die Dreiecksflächenverhältnisse.
3. Berechne die Bahnelemente aus den letzten Werten von $\mathbf{r}_1, \mathbf{r}_3$ und τ_2 .

Die auf diese Weise bestimmte Bahn hat die gesuchte Eigenschaft, daß der auf ihr umlaufende Himmelskörper sich zu den Zeiten t_1 , t_2 und t_3 an den Orten \mathbf{r}_1 , \mathbf{r}_2 und \mathbf{r}_3 befindet, die relativ zur Erde in den Richtungen \mathbf{e}_1 , \mathbf{e}_2 und \mathbf{e}_3 liegen. Damit ist die Aufgabe der Bahnbestimmung zwar im Prinzip gelöst, für die praktische Anwendung weist die Iteration der Sektor- zu Dreiecksverhältnisse aber einige schwerwiegende Beschränkungen auf.

11.2.3 Mehrfache Lösungen

Abgesehen von Ausnahmefällen, in denen die Iteration nicht konvergiert, erhält man mit der vereinfachten Gaußschen Bahnbestimmung immer ein Resultat, das sämtliche Gleichungen des Bahnbestimmungsproblems korrekt erfüllt. Im Zweifelsfall läßt sich dies immer durch entsprechendes Nachrechnen der drei Beobachtungen mit den ermittelten Bahnelementen bestätigen. Dabei ist aber nicht auszuschließen, daß man zum Teil Lösungen erhält, die auf den ersten Blick falsch oder unsinnig erscheinen. Besonders offensichtlich ist dies bei Hyperbelbahnen hoher Exzentrizität, die jeder Erfahrung widersprechen². Die Ursache hierfür liegt darin, daß die Bahnbestimmung aus drei Beobachtungen nicht immer eindeutig ist. Dies ist besonders bei der *Erdbahnlösung* leicht einzusehen. Der Beobachter bewegt sich ja selbst auf einer nahezu ungestörten Keplerbahn um die Sonne und liegt gleichzeitig immer in der vorgegebenen Beobachtungsrichtung. Man erkennt die Erdbahnlösung am schnellsten anhand der Halbachse ($a \approx 1$), der Exzentrizität ($e \approx 0$) und der Neigung gegenüber der Ekliptik ($i \approx 0^\circ$).

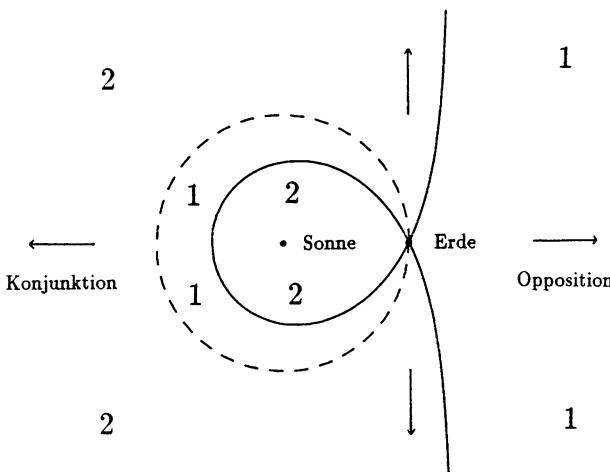


Abb. 11.4. Die Charliersche Grenzlinie (—) trennt die Gebiete, in denen die Bahnbestimmung eindeutig ist (1), von Gebieten der Doppeldeutigkeit (2)

Eine allgemeine Diskussion möglicher Mehrdeutigkeiten bei der Bahnbestimmung ist normalerweise auf kleine Bahnbögen beschränkt. Betrachtet man unter dieser Voraussetzung die gegenseitige Lage von Sonne, Erde und beobachtetem

²Alle bekannten Kometen haben durchweg kleinere Exzentrizitäten als $e = 1.1$.

Himmelskörper, dann lassen sich insgesamt vier verschiedene Gebiete abgrenzen, die durch die Erdbahn und die sogenannte *Charliersche Grenzlinie* (Abb. 11.4) voneinander getrennt sind. Für einen Planetoiden, der um den Zeitpunkt der Opposition herum beobachtet wird, ist das Ergebnis beispielsweise eindeutig, wenn man einmal von der erwähnten Erdbahnlösung absieht. Befindet sich dagegen ein Komet zur Zeit der Beobachtung innerhalb der tropfenförmigen Zone, die die Sonne unmittelbar umgibt, oder im Konjunktionsraum außerhalb der Erdbahn, dann sind prinzipiell zwei Lösungen des Bahnbestimmungsproblems möglich.

11.3 Die vollständige Gauß-Methode

Die in Abschn. 11.2.2 vorgestellte Bahnbestimmungsmethode basiert darauf, Näherungswerte für die Dreiecksflächenverhältnisse n_1 und n_3 durch wiederholtes Einsetzen in die Bahnbestimmungsgleichungen solange zu verbessern, bis alle Gleichungen identisch erfüllt sind. Dabei erhält man jedoch an keiner Stelle eine Information darüber, ob die gewonnene Lösung die einzige Lösung ist, beziehungsweise, wie man bei eventuellen mehrfachen Lösungen die restlichen konstruieren kann.

Einen Überblick über die möglichen Lösungen des Bahnbestimmungsproblems kann man erhalten, wenn man an den Grundgleichungen einige Vereinfachungen vornimmt. Die Lösungen dieses genäherten Problems lassen sich aus den Gauß-Lagrangeschen Gleichungen gewinnen und stimmen im wesentlichen mit den Lösungen der strengen Gleichungen überein. Die Gauß-Lagrangeschen Gleichungen lassen sich schließlich in das alte Rechenschema der Bahnbestimmung einbauen (vgl. Abschn. 11.2.2) und ermöglichen dann in jedem Iterationsschritt eine gezielte Auswahl und Verbesserung der einzelnen Lösungen.

11.3.1 Die Gauß-Lagrangesche Gleichung

Zur Herleitung der Gauß-Lagrangeschen Gleichungen betrachtet man zunächst eine auf Encke zurückgehende, verbesserte Näherung

$$\begin{aligned} n_1 &= \frac{\tau_1}{\tau_2} + \frac{1}{6} \tau_1 \tau_3 \left(1 + \frac{\tau_1}{\tau_2} \right) \cdot \frac{1}{r_2^3} \\ n_3 &= \frac{\tau_3}{\tau_2} + \frac{1}{6} \tau_1 \tau_3 \left(1 + \frac{\tau_3}{\tau_2} \right) \cdot \frac{1}{r_2^3} \end{aligned} \tag{11.29}$$

der Dreiecksflächenverhältnisse. Verglichen mit (11.28) liefert die Enckesche Näherung auch bei größeren Zwischenzeiten gute Ergebnisse, hängt aber von der zunächst nicht bekannten Entfernung r_2 ab.

Setzt man die Enckesche Näherung nun in die Beziehung (11.27) für die geozentrische Entfernung ein, dann erhält man mit den Abkürzungen

$$\begin{aligned} n_{10} &= \frac{\tau_1}{\tau_2} & \mu_1 &= \frac{1}{6} \tau_1 \tau_3 \left(1 + \frac{\tau_1}{\tau_2} \right) \\ n_{30} &= \frac{\tau_3}{\tau_2} & \mu_3 &= \frac{1}{6} \tau_1 \tau_3 \left(1 + \frac{\tau_3}{\tau_2} \right) \end{aligned}$$

sowie

$$\rho_0 = -\frac{1}{D}(n_{10}D_{21} - D_{22} + n_{30}D_{23}) \quad \sigma = +\frac{1}{D}(\mu_1 D_{21} + \mu_3 D_{23})$$

den einfachen Zusammenhang

$$\rho_2 = \rho_0 - \frac{\sigma}{r_2^3} \quad (11.30)$$

zwischen der geozentrischen Entfernung ρ_2 und der heliozentrischen Entfernung r_2 zum Zeitpunkt der zweiten Beobachtung. ρ_0 und σ sind darin bekannte Größen, die sich unmittelbar aus den Beobachtungen ergeben.

Andererseits erhält man einen zweiten Ausdruck für die beiden Größen aus der Gleichung $r_2 = \rho_2 e_2 - \mathbf{R}_2$, die besagt, daß ρ_2 , r_2 und \mathbf{R}_2 die Seiten des Dreiecks Sonne–Erde–Planet sind. Durch Quadrieren ergibt sich daraus die Beziehung

$$r_2 = \sqrt{(\rho_2 - \gamma R_2)^2 + R_2^2(1 - \gamma^2)} \quad , \quad (11.31)$$

in der

$$\gamma = \mathbf{e}_2 \cdot \mathbf{R}_2 / R_2$$

den Cosinus des Winkels zwischen der Beobachtungsrichtung \mathbf{e}_2 und der Sonnenrichtung \mathbf{R}_2 bezeichnet. Zur Zeit der Opposition ist $\gamma = -1$, zur Zeit der Konjunktion ist $\gamma = +1$.

Durch Gleichsetzen der Werte für r_2 aus (11.30) und (11.31) ergibt sich die Gauß-Lagrangesche Gleichung

$$\sqrt[3]{\frac{\sigma}{\rho_0 - \rho_2}} = \sqrt{(\rho_2 - \gamma R_2)^2 + R_2^2(1 - \gamma^2)} \quad , \quad (11.32)$$

die den Schlüssel zur Bestimmung der Unbekannten ρ_2 und r_2 bildet. Dabei kann es insgesamt bis zu drei positive Wertepaare geben, die beide Gleichungen erfüllen. Zu jedem r_2 werden dann die Dreiecksflächenverhältnisse n_1 und n_3 und hiermit ρ_1 und ρ_3 berechnet. Mit dem zugehörigen ρ_2 erhält man daraus die heliozentrischen Orte \mathbf{r}_1 , \mathbf{r}_2 und \mathbf{r}_3 und hat somit die Bahnbestimmung im Rahmen der Enckeschen Näherung gelöst. Da jedes Paar (ρ_2, r_2) auf andere Werte für die heliozentrischen Positionen $\mathbf{r}_{1,2,3}$ führt, kann man insgesamt bis zu drei verschiedene Bahnen erhalten, die die gegebenen Beobachtungen korrekt wiedergeben.

Anstelle des Gleichungssystems (11.30, 11.31) werden in den meisten bekannten Bahnbestimmungsverfahren andere, aber prinzipiell gleichwertige, Formen der Gauß-Lagrangeschen Gleichung verwendet. Gauß selbst entwickelte zur Bestimmung der Ceres-Bahn eine trigonometrische Formulierung, die besonders gut auf das von ihm entwickelte Rechenschema zugeschnitten war. Auf Lagrange geht eine Gleichung achten Grades

$$r_2^8 - \{\rho_0^2 - 2\gamma\rho_0 R_2 + R_2^2\}r_2^6 + \{2\sigma(\rho_0 - \gamma R_2)\}r_2^2 - \{\sigma^2\} = 0 \quad (11.33)$$

für die heliozentrische Entfernung r_2 zurück, die man erhält, wenn man den Wert für ρ_2 aus Gleichung (11.30) in (11.31) einsetzt. Mit der – allerdings nur für sehr

kleine Bahnabschnitte gültigen – Vereinfachung $\sigma \approx \rho_0 R_2^3$ bildet diese Gleichung achten Grades auch den Mittelpunkt der Laplaceschen Bahnbestimmungsmethoden.

Zur Auflösung der Gauß-Lagrangeschen Gleichung ist es hilfreich, zunächst kurz die Graphen der beiden Funktionen

$$\hat{r}(\rho) = \sqrt[3]{\frac{\sigma}{\rho_0 - \rho}} \quad \text{und} \quad \tilde{r}(\rho) = \sqrt{(\rho - \gamma R)^2 + R^2(1 - \gamma^2)}$$

zu betrachten, deren Schnittpunkte die gesuchten Lösungen darstellen. Während die Funktion \hat{r} streng monoton steigend ($\sigma > 0$) oder fallend ($\sigma < 0$) verläuft und bei $\rho = \rho_0$ einen Pol aufweist, stellt die Funktion \tilde{r} eine nach oben geöffnete Hyperbel dar, deren Minimum bei $\rho = \gamma R$ liegt. Hieraus ergibt sich, daß es prinzipiell einen oder drei Schnittpunkte beider Graphen gibt, deren Lage (wegen $R \approx 1$ AE) im einzelnen von σ , ρ_0 und γ abhängt.

Unter der vereinfachenden Annahme $\sigma \approx \rho_0 R^3$, die wie erwähnt aber nur für kleine Bahnbögen gilt, findet man unabhängig von den übrigen Größen immer die Erdbahnlösung mit $\rho = 0$ AE und $r = 1$ AE. Damit verbleiben maximal zwei positive Lösungen ρ , von denen eine die tatsächliche Bahn des beobachteten Objekts repräsentiert. Bezeichnenderweise gilt die obige Annahme aber nicht für das von Gauß behandelte Ceres-Problem, bei dem es lediglich eine Lösung der Gauß-Lagrangeschen Gleichung gibt.

Zur numerischen Behandlung der Gauß-Lagrangeschen Gleichung verwenden wir im weiteren die Funktion `SolveGL`. Sie bestimmt durch eine Kombination von Bisektion und Newtonverfahren alle (bis zu drei) Lösungen von (11.32), scheidet negative, also unphysikalische Lösungen aus und sortiert die übrigen Lösungen nach ihrer Größe. Aufgrund einer Vielzahl notwendiger Fallunterscheidungen und der damit verbundenen Länge des Codes, geben wir an dieser Stelle aber nur die Spezifikation des Funktionskopfes wieder, aus der die einzelnen Aufrufparameter ersichtlich sind.

```
//-----
// SolveGL: Findet die (bis zu drei) Loesungen der Gauss-Lagrange-Gleichung.
//           Gibt nur positive Loesungen zurueck.
// gamma   Cosinus der Sonnenelongation zur Zeit der zweiten Beobachtung
// R        Entfernung Erde-Sonne zur Zeit der zweiten Beobachtung
// rho_0    Nullte Naerherung der geozentrischen Entfernung
// sigma    Hilfsgrössen
// rho_a    Geozentr. Entfernung zur Zeit der zweiten Beobachtung (1. Loesung)
// rho_b    Geozentr. Entfernung zur Zeit der zweiten Beobachtung (2. Loesung)
// rho_c    Geozentr. Entfernung zur Zeit der zweiten Beobachtung (3. Loesung)
// n        Anzahl der gefundenen Loesungen
//-----
void SolveGL ( double gamma, double R, double rho_0, double sigma,
               double& rho_a, double& rho_b, double& rho_c, int& n );
```

11.3.2 Verbesserte Iteration der Dreiecksflächenverhältnisse

Aufgrund der Enckeschen Näherung liefert die Gauß-Lagrangesche Gleichung keine exakten Lösungen des ursprünglichen Bahnbestimmungsproblems. Sie läßt sich

jedoch so mit der Iteration der Dreiecksflächenverhältnisse der gekürzten Gauß-Methode kombinieren, daß sich die jeweiligen Vorteile sinnvoll ergänzen. Man erhält damit das nachfolgend beschriebene Schema der vollständigen Gaußschen Bahnbestimmungsmethode.

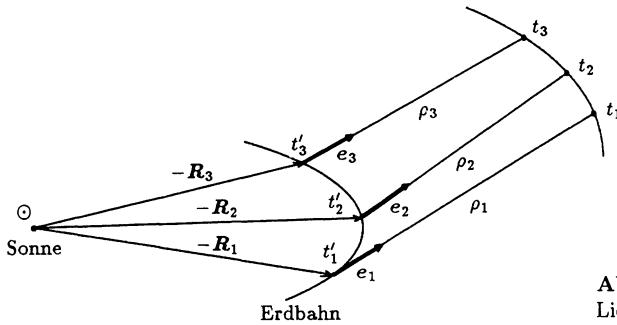
Gegeben seien dazu wieder drei geozentrische Beobachtungen ($\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$), die zugehörigen geozentrischen Koordinaten der Sonne ($\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$) und die Zwischenzeiten (τ_1, τ_2, τ_3). Mit diesen Ausgangsdaten werden die folgenden Rechenschritte durchgeführt:

1. Setze $\mu_1 = (\tau_1\tau_3/6)(1 + \tau_1/\tau_2)$ und $\mu_3 = (\tau_1\tau_3/6)(1 + \tau_3/\tau_2)$ als Ausgangsnäherung für die Korrekturterme der Dreiecksflächenverhältnisse.
2. Wiederhole die Schritte (a)...(e), bis sich μ_1, μ_3 und die übrigen Größen nicht mehr wesentlich verändern.
 - (a) Berechne ρ_0 und σ und löse die zugehörige Gauß-Lagrangische Gleichung. Wähle die erste der drei möglichen Lösungen für ρ_2 aus und berechne die zugehörige heliozentrische Entfernung r_2 .
 - (b) Berechne die Dreiecksflächenverhältnisse $n_1 = n_{10} + \mu_1/r_2^3$ und $n_3 = n_{30} + \mu_3/r_2^3$ und damit die geozentrischen Entfernungen (ρ_1, ρ_2, ρ_3) nach (11.27).
 - (c) Berechne hiermit die heliozentrischen Ortsvektoren ($\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$) aus (11.24).
 - (d) Berechne die Sektor- zu Dreiecksverhältnisse (η_1, η_2, η_3) aus je zwei heliozentrischen Ortsvektoren und der zugehörigen Zwischenzeit nach (11.5).
 - (e) Berechne verbesserte Werte $\mu_1 = (\eta_2/\eta_1 - 1) \cdot (\tau_1/\tau_2) \cdot r_2^3$ und $\mu_3 = (\eta_2/\eta_3 - 1) \cdot (\tau_3/\tau_2) \cdot r_2^3$ für die Korrekturterme der Dreiecksflächenverhältnisse.
3. Berechne die Bahnelemente aus den letzten Werten von $\mathbf{r}_1, \mathbf{r}_3$ und τ_2 .

Weist die Gauß-Lagrangische Gleichung mehr als eine Lösung auf, dann wird das gesamte Schema wiederholt, wobei in Schritt (a) jeweils die zweite beziehungsweise dritte Lösung ausgewählt wird. Anhand von Plausibilitätsüberlegungen können dann gegebenenfalls die Erdbahnlösung oder stark hyperbolische Bahnen als sinnvolle Lösung ausgeschieden werden. Ansonsten hilft in Zweifelsfällen eine vierte Beobachtung unter mehrfachen Lösungen die korrekte Bahn des Kometen oder Planeten zu identifizieren.

11.3.3 Lichtlaufzeit

Für die praktische Anwendung der Methode ist lediglich noch eine kleine Änderung des Rechenschemas erforderlich. Bisher wurde das Bahnbestimmungsproblem so behandelt, als sei die beobachtete Position eines Planeten oder Kometen jederzeit mit der momentanen, geometrischen Position identisch. In Wirklichkeit vergeht



aber zwischen der Aussendung des Lichts und der Beobachtung eine gewisse Zeitspanne, die durch die endliche Lichtgeschwindigkeit

$$\begin{aligned}c &= 173.1 \text{ AE/d} \\1/c &= 0^d 05776/\text{AE}\end{aligned}$$

bestimmt ist. Die Vektoren e_i , die die Beobachtungsrichtungen angeben, weisen deshalb von den heliozentrischen Orten $-\mathbf{R}_i$, an denen sich die Erde zu den Beobachtungszeiten t'_i befindet, zu den drei Orten $\mathbf{r}_i = \rho_i \mathbf{e}_i - \mathbf{R}_i$, an denen der Planet zu etwas früheren Zeitpunkten $t_i < t'_i$ steht (vgl. Abb. 11.5). Die Zeit $t'_i - t_i$, die zwischen der Lichtaussendung und der Beobachtung vergeht, ist dabei gleich der Zeit, die das Licht zum Durchlaufen der Strecke ρ_i benötigt:

$$\rho_i = c \cdot (t'_i - t_i) .$$

Damit entsteht zunächst das Problem, daß die bei der Berechnung der Zwischenzeiten τ_i benötigten Zeitpunkte der Lichtaussendung t_i nicht bekannt sind. Im Verlauf der Bahnbestimmung erhält man aber zunehmend genauere Werte für die geozentrischen Entfernungen (ρ_1, ρ_2, ρ_3), so daß der Einfluß der Lichtlaufzeit problemlos berücksichtigt werden kann. Man verwendet die Beobachtungszeitpunkte t'_i lediglich bei der Bestimmung der Ausgangsnäherung für die Dreiecksflächenverhältnisse. Anschließend werden in jedem Iterationsschritt die Zwischenzeiten

$$\tau_1 = \sqrt{GM_\odot}(t_3 - t_2) \quad \tau_2 = \sqrt{GM_\odot}(t_3 - t_1) \quad \tau_3 = \sqrt{GM_\odot}(t_2 - t_1)$$

mit

$$t_i = t'_i - \rho_i \cdot 0^d 05776/\text{AE} \quad (i = 1 \dots 3) .$$

neu berechnet, sobald die aktuellen Werte der geozentrischen Entfernungen bekannt sind. Die geometrischen Hilfsgrößen \mathbf{d}_i , D_{ij} und D hängen nur von den Richtungsvektoren \mathbf{e}_i und den Sonnenkoordinaten \mathbf{R}_i ab und müssen deshalb nicht korrigiert werden.

11.4 Das Programm GAUSS

Mit dem nun vorgestellten Programm **Gauss** lassen sich die Bahnelemente einer Kometen- oder Asteroidenbahn aus drei bekannten Beobachtungen bestimmen. Aus den gegebenen Werten der Rektaszension und Deklination werden zunächst die Einheitsvektoren der Beobachtungsrichtungen und die Koordinaten der Sonne ermittelt (**Start**). Da sich die gesuchten Bahnelemente wie gewohnt auf die Ekliptik beziehen sollen, werden alle Größen in einem ekliptikalnen Koordinatensystem berechnet, dessen Äquinoktium frei gewählt werden kann. Die eigentliche Bahnbestimmung nach der vollständigen Gauß-Methode erfolgt innerhalb der Funktion **GaussMethod**. Zur Lösung der Gauß-Lagrangeschen Gleichung wird dabei die Funktion **SolveGL** verwendet, die wegen ihres Umfangs aber nicht vollständig abgedruckt ist.

Wegen der Länge der erforderlichen Eingaben werden diese in einer Datei mit Namen **Gauss.dat** zusammengefaßt. Die Datei beginnt mit einer einzelnen Kommentarzeile, die eine leichtere Identifizierung der Daten ermöglichen soll. Die drei folgenden Zeilen enthalten jeweils das Datum in der Form „Jahr, Monat, Tag und Stunde mit Dezimalanteil“, die beobachteten äquatorialen Koordinaten (die Rektaszension in der Form „Stunden, Minuten und Sekunden mit Dezimalanteil“ und die Deklination in der Form „Grad, Minuten und Sekunden mit Dezimalanteil“) sowie einen optionalen Kommentar. Die nächsten beiden Zeilen enthalten das Äquinoktium, auf das sich die beobachteten Positionen beziehen, sowie das gewünschte Äquinoktium der zu berechnenden Bahnelemente. Die folgende Datei enthält als Beispiel einige Beobachtungen des Kleinplaneten Ceres, die Gauß zur Illustration seiner Bahnbestimmungsmethode verwendet hat:

```
Ceres (Beispiel von Gauss)
1805 09 05 24.165   6 23 57.54  22 21 27.08 ! drei Beobachtungen
1806 01 17 22.095   6 45 14.69  30 21 24.20 ! Format: Datum (jmtb),
1806 05 23 20.399   8 07 44.60  28 02 47.04 !           RA (hms), Dec (gms)
1806.0               ! Äquinoktium
1806.0               ! gewünschtes Äquinoktium
```

Bei der Bahnbestimmung wird erwartet, daß alle Beobachtungen in Form von astrometrischen Koordinaten vorliegen, also vom Einfluß derstellaren Aberration befreit sind. Dies ist unter anderem immer dann der Fall, wenn die Koordinaten durch Vergleich mit Umgebungssternen aus einem Sternkatalog gewonnen wurden. Ein Beispiel hierfür ist die Vermessung einer Sternfeldaufnahme, die wohl am häufigsten verwendet wird, um die Bewegung eines Kometen oder Asteroiden zu verfolgen.

```
//-----
// Datei: Gauss.cpp
// Zweck: Bahnbestimmung aus drei Beobachtungen
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//-----
```

```
#include <cmath>
#include <iostream>
```

```

#include <iomanip>
#include <iostream>
#include "APC_Const.h"
#include "APC_IO.h"
#include "APC_Kepler.h"
#include "APC_Math.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"
using namespace std;

//-----
// Start: Lesen der Eingabedatei und Vorverarbeitung der Beobachtungsdaten
//   InputFile  Name der Eingabedatei
//   Header      Kommentar
//   R_Sun[]     Drei Positionsvektoren der Sonne (ekliptikale Koordinaten)
//   e[]         Drei Einheitsvektoren in Richtung des Körpers
//   Mjd0[]     Drei Beobachtungszeiten (MJD)
//   T_eqx      Äquinoxtium von R_Sun und e (in Julianischen Jahrhunderten
//              seit J2000)
//-----
void Start ( char* InputFile,
             char* Header,
             Vec3D R_Sun[], Vec3D e[], double Mjd0[], double& T_eqx )
{
    int      i, Year, Month, Day, deg, min;
    double   Hour, secs, year;
    double   T_eqx0, T, RA, Dec;
    ifstream inp;
    inp.open(InputFile); // Eingabedatei öffnen
    inp.get(Header,81); // Vorspann lesen
    // Beobachtungen lesen
    for (i=0;i<=2;i++) {
        inp >> Year >> Month >> Day >> Hour;
        inp >> deg >> min >> secs;
        RA = Rad*15.0*Ddd(deg,min,secs);
        inp >> deg >> min >> secs;
        Dec = Rad*Ddd(deg,min,secs);
        inp.ignore(81,'\n');
        Mjd0[i] = Mjd(Year,Month,Day) + Hour/24.0;
        e[i]    = Vec3D(Polar(RA,Dec));
    }
    // Äquinoxtium der Beobachtungen und gesuchtes Äquinoxtium
    inp >> year; inp.ignore(81,'\n'); T_eqx0 = (year-2000.0)/100.0;
    inp >> year; inp.ignore(81,'\n'); T_eqx = (year-2000.0)/100.0;
    // Ausgangsdaten für die Bahnbestimmung
    // (Beobachtungen und Sonnenposition bezogen auf Ekliptik und
    // Äquinoxtium T_eqx)
    for (i=0;i<=2;i++) {
        T          = (Mjd0[i]-MJD_J2000) / 36525.0;
        e[i]       = PrecMatrix_Ecl(T_eqx0,T_eqx) * Equ2EclMatrix(T_eqx0) * e[i];
        R_Sun[i]  = PrecMatrix_Ecl(T,T_eqx) * SunPos(T);
    }
}

```

```

// Uebersicht der Ausgangsdaten anzeigen
cout << " Uebersicht der Bahnbestimmungs-Ausgangsdaten fuer" << endl << endl
    << "   " << Header << endl << endl
    << "   (geozentrische ekliptikale Koordinaten; Aequinoktium J"
    << fixed << setprecision(2) << 100.0*T_eqx+2000.0 << ")" << endl
    << endl;
cout << "   Beobachtung"
    << setw(25) << "1" << setw(12) << "2" << setw(12) << "3" << endl;
cout << "   Julianisches Datum           ";
for (i=0;i<=2;i++) { cout << setw(12) << Mjd0[i]+2400000.5; }
cout << endl;
cout << "   Sonnenlaenge [Grad]          " << setprecision(4);
for (i=0;i<=2;i++) { cout << setw(12) << Deg*(R_Sun[i][phi]); }
cout << endl;
cout << "   Laenge Planet/Komet [Grad]   ";
for (i=0;i<=2;i++) { cout << setw(12) << Deg*((e[i])[phi]); }
cout << endl;
cout << "   Breite Planet/Komet [Grad]  ";
for (i=0;i<=2;i++) { cout << setw(12) << Deg*((e[i])[theta]); }
cout << endl;
}

//-----
// DumpElem: Formatierte Ausgabe der Bahnelemente
// Mjd_p      Zeitpunkt des Periheldurchgangs als MJD
// q          Periheldistanz in [AE]
// e          Exzentrizitaet
// i          Bahnneigung in [rad]
// Omega      Laenge des aufsteigenden Knotens in [rad]
// omega      Argument des Perihels in [rad]
// T_eqx     Bezugsaequinoktium (Julianische Jahrhunderte seit J2000)
//-----
void DumpElem ( double Mjd_p, double q, double e, double i,
                double Omega, double omega, double T_eqx )
{
    ...
}

//-----
// Retard: Beruecksichtigung der Lichtlaufzeit und Berechnung der
//          Zeitdifferenzen
// Mjd0[]     Beobachtungszeiten (t1',t2',t3') (MJD)
// rho[]      Drei geozentrische Entfernungen in [AE]
// Mjd[]      Zeitpunkte der Lichtaussendung (t1,t2,t3) (MJD)
// tau[]      Skalierte Zeitdifferenzen
//-----
void Retard ( const double Mjd0[], const double rho[],
              double Mjd[], double tau[] )
{
    for (int i=0;i<=2;i++) Mjd[i] = Mjd0[i] - rho[i]/c_light;
    tau[0] = kGauss * (Mjd[2]-Mjd[1]);
    tau[1] = kGauss * (Mjd[2]-Mjd[0]);
    tau[2] = kGauss * (Mjd[1]-Mjd[0]);
}

```

```

//-----
// SolveGL: Findet die (bis zu drei) Loesungen der Gauss-Lagrange-Gleichung.
// Gibt nur positive Loesungen zurueck.
// gamma Cosinus der Sonnenelongation zur Zeit der zweiten Beobachtung
// R Entfernung Erde-Sonne zur Zeit der zweiten Beobachtung
// rho_0 Nullte Naerherung der geozentrischen Entfernung
// sigma Hilfsgroesse
// rho_a Geozentr. Entfernung zur Zeit der zweiten Beobachtung (1. Loesung)
// rho_b Geozentr. Entfernung zur Zeit der zweiten Beobachtung (2. Loesung)
// rho_c Geozentr. Entfernung zur Zeit der zweiten Beobachtung (3. Loesung)
// n Anzahl der gefundenen Loesungen
//-----
void SolveGL ( double gamma, double R, double rho_0, double sigma,
               double& rho_a, double& rho_b, double& rho_c, int& n )
{
    ...
}

//-----
// Select: Auswahl einer einzelnen Loesung der Gauss-Lagrange-Gleichung
// rho_a 1. Loesung
// rho_b 2. Loesung
// rho_c 3. Loesung
// n_sol Gesamtanzahl der Loesungen
// n Nummer der ausgewahlten Loesung
// <return> Ausgewaehlte Loesung rho
//-----
double Select ( double rho_a, double rho_b, double rho_c,
                 int n_sol, int n )
{
    if ( (n<1) || (n>n_sol) ) {
        cerr << " Fehler in Select: n = " << setw(2) << n << " n_sol = "
            << setw(2) << n_sol << endl;
        exit(1);
    }
    else
        switch(n) {
            case 1: return rho_a;
            case 2: return rho_b;
            case 3: return rho_c;
        };
    return 0.0;
}

//-----
// GaussMethod: Bahnbestimmung nach dem Verfahren von Gauss
// R_Sun[] Drei Positionsvektoren der Sonne (ekliptikale Koordinaten)
// e[] Drei Einheitsvektoren in Richtung des Koerpers
// Mjd0[] Drei Beobachtungszeiten (MJD)
// N_run Nummer des Durchlaufs (entspricht der gewaehlten Loesung der
// Gauss-Lagrange-Gleichung)
// N_sol Gesamtzahl der Loesungen der Gauss-Lagrange-Gleichung
// Mjd_p Zeitpunkt des Periheldurchgangs als MJD
// q Periheldistanz in [AE]
// ecc Exzentrizitaet

```

```

// inc      Bahnneigung in [rad]
// Omega    Laenge des aufsteigenden Knotens in [rad]
// omega    Argument des Perihels in [rad]
//-----

void GaussMethod ( const Vec3D R_Sun[], const Vec3D e[], const double Mjd0[],
                   int& N_run, int& N_sol,
                   double& Mjd_p, double& q, double& ecc,
                   double& inc, double& Omega, double& omega )
{
    // Konstanten
    const double eps_rho = 1.0e-8;
    const int maxit = 20;
    // Variablen
    int i,j,iterat;
    double rho_old;
    double Det,l,L,gamma;
    double rho_max,rho_mean,rho_min;
    double Mjd[3],rho[3],n[3],n0[3],tau[3],eta[3];
    double mu[3];
    Vec3D d[3],r[3];
    double D[3][3];

    // Matrix D und Determinante Det
    d[0] = Cross(e[1],e[2]);
    d[1] = Cross(e[2],e[0]);
    d[2] = Cross(e[0],e[1]);
    for (i=0;i<=2;i++) for (j=0;j<=2;j++) D[i][j] = Dot(d[i],R_Sun[j]);
    Det = Dot(e[2],d[2]);
    // Richtungskosinus des Einheitsvektors in Beobachtungsrichtung bezogen
    // auf die Richtung zur Sonne zum Zeitpunkt der zweiten Beobachtung
    gamma = Dot(e[1],R_Sun[1])/Norm(R_Sun[1]);
    // Zeitdifferenzen tau[i] und erste Naherungen fuer mu[0] und mu[2]
    tau[0] = kGauss*(Mjd0[2]-Mjd0[1]);
    tau[1] = kGauss*(Mjd0[2]-Mjd0[0]);
    tau[2] = kGauss*(Mjd0[1]-Mjd0[0]);
    mu[0] = (1.0/6.0) * tau[0]*tau[2] * (1.0+tau[0]/tau[1]);
    mu[2] = (1.0/6.0) * tau[0]*tau[2] * (1.0+tau[2]/tau[1]);
    // Vorspann
    cout << endl << endl
        << " Loesung Nr." << setw(1) << N_run << endl << endl
        << " Loesungen (rho_2 in [AE]) der Gauss-Lagrangeschen Gleichung "
        << endl << endl;
    cout << " Iteration           Anzahl   1.Loesung   2.Loesung   3.Loesung"
        << endl;

    // Iterative Verbesserung von tau[i] und mu[i]
    rho[1] = 0.0;
    iterat = 0;
    do {

        // Vorhergehenden Wert sichern und Iterationszaehler erhöhen
        rho_old = rho[1];
        iterat++;
}

```

```

// Bestimme die geozentrische Entfernung rho zur Zeit der zweiten
// Beobachtung aus der Gauss-Lagrange-Gleichung
n0[0] = tau[0]/tau[1];
n0[2] = tau[2]/tau[1];
L = - ( n0[0]*D[1][0]-D[1][1]+n0[2]*D[1][2] ) / Det;
l = ( mu[0]*D[1][0] + mu[2]*D[1][2] ) / Det;
SolveGL (gamma, Norm(R_Sun[1]), L, l, rho_min, rho_mean, rho_max, N_sol);
rho[1] = Select ( rho_min, rho_mean, rho_max, N_sol, N_run );
r[1] = rho[1]*e[1] - R_Sun[1];
cout << setw(4) << iterat << setw(23) << N_sol << fixed << setprecision(8)
<< setw(15) << rho_min << setw(12) << rho_mean
<< setw(12) << rho_max << endl;

// Berechne n1 und n3
n[0] = n0[0] + mu[0]/pow(Norm(r[1]),3);
n[2] = n0[2] + mu[2]/pow(Norm(r[1]),3);
// Geozentrische Entfernungen rho_1 und rho_3 aus n_1 und n_3
rho[0] = ( n[0]*D[0][0] - D[0][1] + n[2]*D[0][2] ) / (n[0]*Det);
rho[2] = ( n[0]*D[2][0] - D[2][1] + n[2]*D[2][2] ) / (n[2]*Det);
// Beruecksichtige die Lichtlaufzeit und berechne skalierte Zeitdifferenzen
Retard ( Mjd0, rho, Mjd, tau );
// Heliozentrische Orte
for (i=0;i<=2;i++)
    r[i] = rho[i]*e[i] - R_Sun[i];

// Sektor/Dreiecks-Verhaeltnisse eta_i
eta[0] = FindEta ( r[1], r[2], tau[0] );
eta[1] = FindEta ( r[0], r[2], tau[1] );
eta[2] = FindEta ( r[0], r[1], tau[2] );
// Verbesserte Werte fuer mu_1 und mu_3
mu[0] = ( eta[1]/eta[0] - 1.0 ) * (tau[0]/tau[1]) * pow(Norm(r[1]),3);
mu[2] = ( eta[1]/eta[2] - 1.0 ) * (tau[2]/tau[1]) * pow(Norm(r[1]),3);
if (maxit<=iterat) {
    cerr << " (Konvergenzprobleme; Iteration nach "
        << setw(2) << maxit << " Schritten abgebrochen)"
        << endl;
    break;
}
while (fabs(rho[1]-rho_old) > eps_rho);

cout << endl
    << " Geozentrische und heliozentrische Entfernungen" << endl
    << endl;
cout << " Beobachtung"
    << setw(25) << "1" << setw(12) << "2" << setw(12) << "3" << endl;
cout << " rho [AE]" << setw(20) << " " << fixed << setprecision(8);
for (i=0;i<=2;i++) cout << setw(12) << rho[i]; cout << endl;
cout << " r [AE] " << setw(20) << " " << fixed << setprecision(8);
for (i=0;i<=2;i++) cout << setw(12) << Norm(r[i]); cout << endl;

// Ermittle die Bahnelemente aus der ersten und der dritten Beobachtung
Elements ( GM_Sun, Mjd[0], Mjd[2], r[0], r[2],
            Mjd_p, q, ecc, inc, Omega, omega );
}

```

```
-----  
//  
// Hauptprogramm  
//  
-----  
  
void main(int argc, char* argv[]){  
    // Variablen  
    char Header[81];  
    int N_run, N_sol;  
    double T_eqx;  
    double Mjd0[3];  
    Vec3D R_Sun[3], e[3];  
    double Mjd_p, q, ecc, inc, Omega, omega;  
    char InputFile[APC_MaxFilename] = "";  
    char OutputFile[APC_MaxFilename] = "";  
    bool FoundInputfile = false;  
    bool FoundOutputfile = false;  
    ofstream OutFile;  
  
    // Titel  
    cout << endl  
        << "      GAUSS: Bahnbestimmung aus drei Beobachtungen"  
        << endl  
        << "      (c) 1999 Oliver Montenbruck, Thomas Pfleger"  
        << endl << endl;  
    // Ermittle die Namen der Ein- und Ausgabedatei  
    GetFilenames( argc, argv, "Gauss.dat", InputFile, FoundInputfile,  
                  OutputFile, FoundOutputfile );  
    // Abbruch, falls die Eingabedatei nicht gefunden wurde  
    if (!FoundInputfile) { cerr << " Abbruch." << endl; exit(-1); }  
    // Bei Bedarf die Ausgabe in eine Datei umlenken  
    if (FoundOutputfile) {  
        OutFile.open(OutputFile); if (OutFile.is_open()) cout = OutFile;  
    }  
  
    // Initialisierung  
    Start ( InputFile, Header, R_Sun, e, Mjd0, T_eqx );  
  
    // Alle Loesungen behandeln  
    N_run = 0;  
    N_sol = 1;  
    do {  
        N_run++;  
        // Bestimme die Bahn mit dem Gauss'schen Verfahren fuer die Loesung  
        // N_run der Gauss-Lagrange-Gleichung  
        GaussMethod ( R_Sun, e, Mjd0, N_run, N_sol, Mjd_p,q,ecc,inc,Omega,omega );  
        // Ausgabe der Bahnelemente  
        DumpElem ( Mjd_p,q,ecc,inc,Omega,omega, T_eqx );  
    }  
    while (N_run<N_sol);  
  
    if (OutFile.is_open()) OutFile.close();  
}
```

Nach dem Aufruf von Gauss werden vom Benutzer keine Eingaben erwartet, da alle notwendigen Informationen bereits in der Datei Gauss.dat enthalten sind. Für die oben angegebenen Zahlenwerte erhält man zum Beispiel das folgende Protokoll:

GAUSS: Bahnbestimmung aus drei Beobachtungen
 (c) 1999 Oliver Montenbruck, Thomas Pfleger

Uebersicht der Bahnbestimmungs-Ausgangsdaten fuer

Ceres (Beispiel von Gauss)
 (geozentrische ekliptikale Koordinaten; Aequinoktium J1806.00)

Beobachtung	1	2	3
Julianisches Datum	2380570.51	2380704.42	2380830.35
Sonnenlaenge [Grad]	162.9166	297.2126	61.9405
Laenge Planet/Komet [Grad]	95.5389	99.8239	118.1018
Breite Planet/Komet [Grad]	-0.9918	7.2775	7.6473

Loesung Nr. 1

Loesungen (rho_2 in [AE]) der Gauss-Lagrangeschen Gleichung

Iteration	Anzahl	1.Loesung	2.Loesung	3.Loesung
1	1	1.63338760	0.00000000	0.00000000
2	1	1.63476170	0.00000000	0.00000000
3	1	1.63599887	0.00000000	0.00000000
4	1	1.63655182	0.00000000	0.00000000
5	1	1.63676833	0.00000000	0.00000000
6	1	1.63684968	0.00000000	0.00000000
7	1	1.63687981	0.00000000	0.00000000
8	1	1.63689091	0.00000000	0.00000000
9	1	1.63689499	0.00000000	0.00000000
10	1	1.63689649	0.00000000	0.00000000
11	1	1.63689704	0.00000000	0.00000000
12	1	1.63689724	0.00000000	0.00000000
13	1	1.63689731	0.00000000	0.00000000
14	1	1.63689734	0.00000000	0.00000000
15	1	1.63689735	0.00000000	0.00000000

Geozentrische und heliozentrische Entfernungen

Beobachtung	1	2	3
rho [AE]	2.90181811	1.63689735	2.95876464
r [AE]	2.68083076	2.58786985	2.54398416

Bahnelemente (Aequinoktium J1806.00)

Periheldurchgang	tp	1806/06/28.039	(JD 2380865.539)
Periheldistanz	q [AE]	2.541676	
Grosse Halbachse	a [AE]	2.767165	
Exzentrizitaet	e	0.081487	
Bahnneigung	i	10.6178	Grad
Aufsteigender Knoten	Omega	80.9788	Grad
Laenge des Perihels	pi	147.0173	Grad
Argument des Perihels	omega	66.0385	Grad

Wie man an den ekliptikalnen Koordinaten der Sonne und der Ceres erkennen kann, befand sich der Kleinplanet zur Zeit der Beobachtung in der Nähe der Opposition. Die Gefahr einer doppelten Lösung der Bahnbestimmung besteht bei den hier vorliegenden Daten also nicht.

Als Beispiel einer Bahnbestimmung mit zwei nichttrivialen, positiven Lösungen der Gauß-Lagrangeschen Gleichung betrachten wir deshalb noch drei Beobachtungen des Kometen Orkisz (1925 I) vom April 1925:

```
Komet Orkisz (Beispiel aus Stracke: Bahnbestimmung, S.182)
1925 04 05 2.786 22 26 43.51 16 37 16.00 ! drei Beobachtungen
1925 04 08 2.731 22 29 42.90 19 46 25.10 ! Format: Datum (jmth),
1925 04 11 2.614 22 32 55.00 23 04 52.30 ! RA (hms), Dec (gms)
1925.0 ! Aequinoktium
1925.0 ! gewuenschtes Aequinoktium
```

Startet man das Programm Gauss unter zusätzlicher Angabe der Eingabedatei Orkisz.dat (vgl. Abschn. A.1.3), dann erhält man das folgende Ergebnis:

Uebersicht der Bahnbestimmungs-Ausgangsdaten fuer

Komet Orkisz (Beispiel aus Stracke: Bahnbestimmung, S.182)

(geozentrische ekliptikale Koordinaten; Aequinoktium J1925.00)

Beobachtung	1	2	3
Julianisches Datum	2424245.62	2424248.61	2424251.61
Sonnenlaenge [Grad]	14.8175	17.7642	20.7031
Laenge Planet/Komet [Grad]	345.0988	347.2423	349.5855
Breite Planet/Komet [Grad]	24.4159	27.0070	29.6985

Loesung Nr.1

Loesungen (rho_2 in [AE]) der Gauss-Lagrangeschen Gleichung

Iteration	Anzahl	1.Loesung	2.Loesung	3.Loesung
1	2	1.67288439	6.37592623	-0.00141974
2	2	1.67314456	6.37552347	-0.00162268
3	2	1.67314511	6.37552331	-0.00162312
4	2	1.67314510	6.37552331	-0.00162312
5	2	1.67314511	6.37552331	-0.00162313
6	2	1.67314510	6.37552331	-0.00162312
7	2	1.67314509	6.37552331	-0.00162311

Geozentrische und heliozentrische Entfernungen

Beobachtung	1	2	3
rho [AE]	1.71490811	1.67314509	1.63322160
r [AE]	1.10821587	1.10918577	1.11233811

Bahnelemente (Aequinoktium J1925.00)

Periheldurchgang	tp	1925/04/05.280	(JD 2424245.780)
Periheldistanz	q [AE]	1.108212	

Grosse Halbachse	a [AE]	-80.901263
Exzentritzaet	e	1.013698
Bahnneigung	i	101.2244 Grad
Aufsteigender Knoten	Omega	318.9892 Grad
Laenge des Perihels	pi	359.8990 Grad
Argument des Perihels	omega	40.9098 Grad

Loesung Nr. 2

Loesungen (ρ_2 in [AE]) der Gauss-Lagrangeschen Gleichung

Iteration	Anzahl	1. Loesung	2. Loesung	3. Loesung
1	2	1.67288439	6.37592623	-0.00141974
2	2	1.67368234	6.37178371	-0.00197746
3	2	1.67368197	6.37178566	-0.00197720
4	2	1.67368197	6.37178566	-0.00197720

Geozentrische und heliozentrische Entfernungen

Beobachtung	1	2	3
rho [AE]	6.53717259	6.37178566	6.22572525
r [AE]	5.77840936	5.63979853	5.52113670

Bahnelemente (Aequinoktium J1925.00)

Periheldurchgang	tp	1925/04/25.025	(JD 2424265.525)
Periheldistanz	q [AE]	5.262900	
Grosse Halbachse	a [AE]	-0.020764	
Exzentritzaet	e	254.460587	
Bahnneigung	i	67.3802 Grad	
Aufsteigender Knoten	Omega	326.7908 Grad	
Laenge des Perihels	pi	21.6630 Grad	
Argument des Perihels	omega	54.8721 Grad	

Die Gauß-Lagrangesche Gleichung hat hier insgesamt drei Lösungen, von denen eine ($\rho_2 \approx 0.0$ AE) die Erdbahnlösung repräsentiert. Die übrigen Lösungen ($\rho_2 \approx 1.7$ AE) und ($\rho_2 \approx 6.4$ AE) stellen zunächst beide physikalisch mögliche Kometenbahnen dar. Aus Plausibilitätsgründen ($e \gg 1$) kann die zweite Lösung jedoch als unwahrscheinlich ausgeschieden werden.

12. Astrometrie

Die Fotografie bietet eine verhältnismäßig einfache Möglichkeit, die Koordinaten eines Kometen oder Planeten durch den Vergleich mit bekannten Sternpositionen zu bestimmen. Neben der Aufnahme selbst benötigt man dazu lediglich einen Sternkatalog und eventuell einen Atlas zur leichteren Identifizierung der Umgebungssterne.

Zum besseren Verständnis der folgenden Überlegungen ist in Abb. 12.1 eine Aufnahme des Kometen Bradfield von 1982 skizziert. Der Vergleich mit einem Sternatlas zeigt, daß das Bild ein Gebiet von $\pm 4^m$ Rektaszension und $\pm 1^\circ$ Deklination erfaßt und um den Punkt ($\alpha_{2000} = 12^h 28^m 45^s$, $\delta_{2000} = 44^\circ 00'$) zentriert ist. Neben dem Kometen lassen sich eine Reihe von Sternen identifizieren, deren Koordinaten dem *Position and Proper Motion Catalogue* (PPM) entnommen werden können. Dieser überdeckt den gesamten Nord- und Südhimmel und enthält die Positionen von rund 470 000 Sternen. Die mittlere Entfernung zweier benachbarter Katalogsterne beträgt dabei etwa ein halbes Grad. Mindestens drei Vergleichssterne mit bekannten Koordinaten werden benötigt, um den Ort eines Himmelskörpers aus einer Fotografie bestimmen zu können. Um möglichst genaue Resultate zu erzielen, wird man allerdings versuchen, die gesuchten Koordinaten mit Hilfe möglichst vieler Umgebungssterne zu ermitteln. Hierzu soll zunächst erläutert werden, wie ein Ausschnitt des Himmels mit Sternen verschiedener Rektaszension und Deklination auf einer Aufnahme wiedergegeben wird.

12.1 Die fotografische Abbildung

Ein ideales Kamera- oder Fernrohrobjectiv vereinigt die von einem Stern ausgehenden Lichtstrahlen in einem Punkt der Filmebene. Diese liegt in einer Entfernung F hinter dem Objektiv, die als Brennweite bezeichnet wird. Der Punkt P , auf den der Stern abgebildet wird, läßt sich konstruieren, indem man den Lichtstrahl durch die Objektivmitte O mit der Filmebene schneidet (Abb. 12.2). In dem von \mathbf{u} , \mathbf{v} und \mathbf{w} aufgespannten Koordinatensystem gibt

$$\mathbf{e} = \begin{pmatrix} \cos(\delta) \cos(\alpha - \alpha_0) \\ \cos(\delta) \sin(\alpha - \alpha_0) \\ \sin(\delta) \end{pmatrix}$$

die Richtung zu einem Stern der Rektaszension α und der Deklination δ an. Entsprechend definiert

$$\mathbf{e}_0 = \begin{pmatrix} \cos(\delta_0) \\ 0 \\ \sin(\delta_0) \end{pmatrix}$$

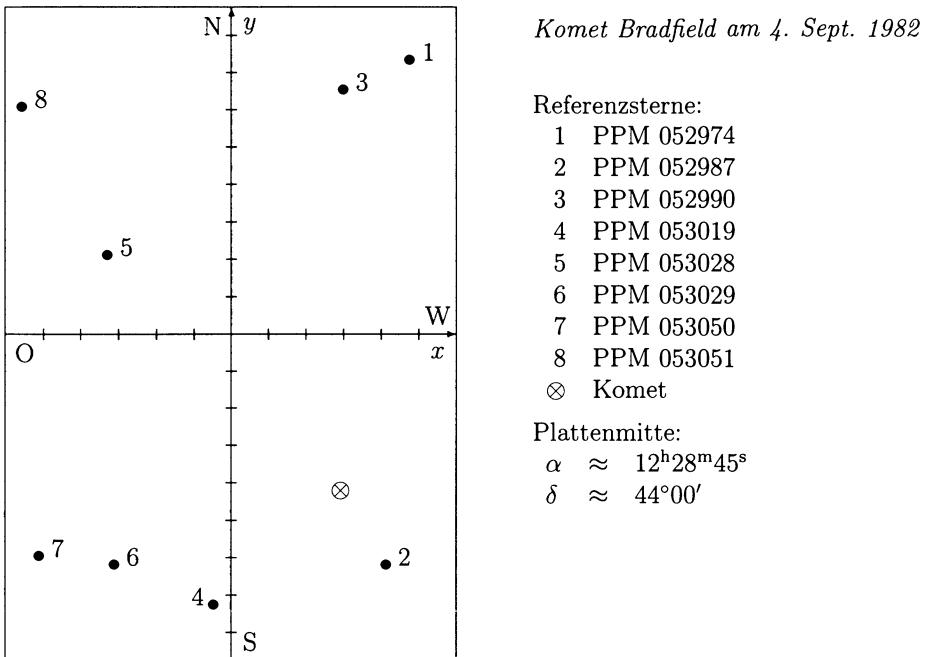


Abb. 12.1. Beispiel einer Sternfeldaufnahme

den Ort (α_0, δ_0) am Himmel, auf den die optische Achse der Kamera ausgerichtet ist. Die Vektoren $\mathbf{F} = -F \cdot \mathbf{e}_0$ und $\mathbf{p} = -p \cdot \mathbf{e}$ beschreiben den Lichtweg vom Objektiv zur Plattenmitte und zum Bildpunkt P des Sterns. Sie schließen miteinander den Winkel φ ein mit

$$\begin{aligned}\cos(\varphi) &= \mathbf{e}_0 \cdot \mathbf{e} \\ &= \cos(\delta_0) \cos(\delta) \cos(\alpha - \alpha_0) + \sin(\delta_0) \sin(\delta) \quad .\end{aligned}$$

Innerhalb der Filmebene wird durch die Vektoren

$$\mathbf{e}_X = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \text{und} \quad \mathbf{e}_Y = \begin{pmatrix} +\sin(\delta_0) \\ 0 \\ -\cos(\delta_0) \end{pmatrix}$$

ein Koordinatensystem zur Vermessung der Aufnahme definiert, das in Nord-Süd- und Ost-West-Richtung orientiert ist. Mit den Koordinaten X und Y des Bildpunktes P , die in Einheiten der Brennweite F gemessen werden, lässt sich \mathbf{p} in der folgenden Form darstellen:

$$\mathbf{p} = \mathbf{F} + (F \cdot X) \cdot \mathbf{e}_X + (F \cdot Y) \cdot \mathbf{e}_Y \quad .$$

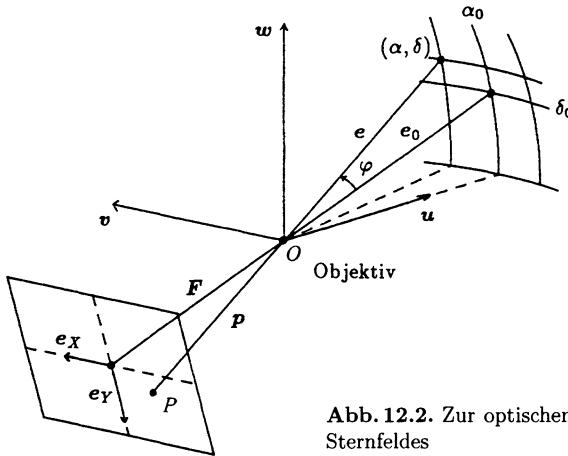


Abb. 12.2. Zur optischen Abbildung bei der Fotografie eines Sternfeldes

Schreibt man diese Gleichung komponentenweise an, dann erhält man für den Zusammenhang von (α, δ) einerseits und (X, Y) andererseits die drei Beziehungen

$$\begin{aligned} p \cos(\delta) \cos(\alpha - \alpha_0) &= F \cos(\delta_0) - FY \sin(\delta_0) \\ p \cos(\delta) \sin(\alpha - \alpha_0) &= -FX \\ p \sin(\delta) &= F \sin(\delta_0) + FY \cos(\delta_0) \end{aligned} \quad (12.1)$$

mit

$$p = |\mathbf{p}| = F \sqrt{1 + X^2 + Y^2}$$

oder

$$\begin{aligned} p &= F / \cos(\varphi) \\ &= F / (\cos(\delta_0) \cos(\delta) \cos(\alpha - \alpha_0) + \sin(\delta_0) \sin(\delta)) . \end{aligned}$$

Aufgelöst nach den sphärischen Koordinaten führt dies auf die Gleichungen

$$\begin{aligned} \alpha &= \alpha_0 + \arctan \left\{ \frac{-X}{\cos(\delta_0) - Y \sin(\delta_0)} \right\} \\ \delta &= \arcsin \left\{ \frac{\sin(\delta_0) + Y \cos(\delta_0)}{\sqrt{1 + X^2 + Y^2}} \right\} . \end{aligned} \quad (12.2)$$

Die Umkehrung von (12.2) ergibt sich ebenfalls aus (12.1) zu

$$\begin{aligned} X &= -\frac{\cos(\delta) \sin(\alpha - \alpha_0)}{\cos(\delta_0) \cos(\delta) \cos(\alpha - \alpha_0) + \sin(\delta_0) \sin(\delta)} \\ Y &= -\frac{\sin(\delta_0) \cos(\delta) \cos(\alpha - \alpha_0) - \cos(\delta_0) \sin(\delta)}{\cos(\delta_0) \cos(\delta) \cos(\alpha - \alpha_0) + \sin(\delta_0) \sin(\delta)} . \end{aligned} \quad (12.3)$$

Diese beiden Transformationen lassen sich in einfacher Weise als Funktionen formulieren:

```

//-----
// StdEqu: Berechnung von aequatorialen Koordinaten aus Standardkoordinaten,
// wie sie beim Ausmessen von Fotoplatten verwendet werden
// RAO      Rektaszension der optischen Achse [rad]
// Dec0     Deklination der optischen Achse [rad]
// X        Standardkoordinate X [dimensionslos]
// Y        Standardkoordinate Y [dimensionslos]
// RA       Rektaszension [rad]
// Dec      Deklination [rad]
//-----
void StdEqu ( double RAO, double Dec0, double X, double Y,
              double& RA, double& Dec )
{
    RA = RAO + atan ( -X / (cos(Dec0)-Y*sin(Dec0)) );
    Dec = asin ( (sin(Dec0)+Y*cos(Dec0))/sqrt(1.0+X*X+Y*Y) );
}
//-----
// EquStd: Berechnung von Standardkoordinaten aus aequatorialen Koordinaten
// RAO      Rektaszension der optischen Achse [rad]
// Dec0     Deklination der optischen Achse [rad]
// RA       Rektaszension [rad]
// Dec      Deklination [rad]
// X        Standardkoordinate X [dimensionslos]
// Y        Standardkoordinate Y [dimensionslos]
//-----
void EquStd ( double RAO, double Dec0, double RA, double Dec,
              double& X, double& Y )
{
    const double c =
        cos(Dec0)*cos(Dec)*cos(RA-RAO)+sin(Dec0)*sin(Dec);
    X = - ( cos(Dec)*sin(RA-RAO) ) / c;
    Y = - ( sin(Dec0)*cos(Dec)*cos(RA-RAO)-cos(Dec0)*sin(Dec) ) / c;
}

```

12.2 Die Plattenkonstanten

Die dimensionslosen Koordinaten X und Y werden als Standardkoordinaten bezeichnet, weil sie definitionsgemäß nicht von der Brennweite der verwendeten Optik abhängen und sich auf ein Koordinatensystem beziehen, das streng parallel zum Meridian durch den Mittelpunkt (α_0, δ_0) der Aufnahme ausgerichtet ist. Legt man dieses Koordinatensystem auch der Auswertung der Fotografie zugrunde, dann muß man die gemessenen Koordinaten x und y lediglich durch die Brennweite dividieren, um die Standardkoordinaten zu erhalten:

$$X = x/F \quad Y = y/F .$$

Im allgemeinen wird der Ursprung des verwendeten Koordinatensystems allerdings nicht genau mit dem Schnittpunkt der optischen Achse und der Filmebene zusammenfallen. Eine solche Verschiebung um die Strecke $(\Delta x, \Delta y)$ kann jedoch durch eine kleine Modifikation der obigen Gleichung berücksichtigt werden:

$$X = x/F - (\Delta x)/F \quad Y = y/F - (\Delta y)/F .$$

Sind darüber hinaus die Koordinatenachsen um einen Winkel γ gegen die Nord-Süd-Richtung verdreht, dann können die Umrechnungsformeln wie folgt erweitert werden:

$$\begin{aligned} X &= (x \cdot \cos(\gamma) - y \cdot \sin(\gamma))/F - (\Delta x)/F \\ Y &= (x \cdot \sin(\gamma) + y \cdot \cos(\gamma))/F - (\Delta y)/F \quad . \end{aligned}$$

Die Verschiebung und Verdrehung des Koordinatensystems ist jedoch nicht der einzige Faktor, der den Zusammenhang zwischen den gemessenen Koordinaten und den Standardkoordinaten bestimmt. Fehler der Optik sowie eine mögliche Verkippung oder Verformung des Films machen im allgemeinen weitere Korrekturen erforderlich. Man verwendet deshalb einen allgemeinen Ansatz der Form

$$\begin{aligned} X &= a \cdot x + b \cdot y + c \\ Y &= d \cdot x + e \cdot y + f \end{aligned} \tag{12.4}$$

mit sechs sogenannten *Plattenkonstanten* a, b, c, d, e und f zur gegenseitigen Umrechnung der Koordinatenpaare (x, y) und (X, Y) . Die Plattenkonstanten werden dabei nicht von vornherein als bekannt angenommen, sondern mit Hilfe der Referenzsterne bestimmt. Kennt man nämlich die äquatorialen Koordinaten $(\alpha_i, \delta_i)_{i=1,2,3}$ dreier Vergleichssterne, dann folgen daraus mit Hilfe von (12.3) die zugehörigen Standardkoordinaten $(X_i, Y_i)_{i=1,2,3}$. Zusammen mit den gemessenen Koordinaten $(x_i, y_i)_{i=1,2,3}$ ergibt dies die drei Gleichungen

$$\begin{aligned} X_1 &= x_1 \cdot a + y_1 \cdot b + c \\ X_2 &= x_2 \cdot a + y_2 \cdot b + c \\ X_3 &= x_3 \cdot a + y_3 \cdot b + c \quad , \end{aligned} \tag{12.5}$$

die sich durch verschiedene Umformungen nach a, b und c auflösen lassen. Aus den Gleichungen für Y_i erhält man ganz entsprechend die übrigen Plattenkonstanten d, e und f . Die Brennweite der Aufnahmeoptik wird dabei nicht mehr benötigt. Dies ist besonders dann von Vorteil, wenn man eine Vergrößerung der Originalaufnahme bearbeitet, deren Maßstab nicht bekannt ist.

Wie man sieht, können die Plattenkonstanten durch Lösung zweier Gleichungssysteme mit je drei Unbekannten bestimmt werden, wenn mindestens drei Referenzsterne bekannter Rektaszension und Deklination zur Verfügung stehen. Da die Messung der Sternkoordinaten auf der Aufnahme aber nie völlig fehlerfrei sein kann, ist es wünschenswert, möglichst viele bekannte Sterne in die Ermittlung der Plattenkonstanten einzubeziehen. Allerdings ist ein Gleichungssystem der Form

$$\begin{aligned} X_1 &= x_1 \cdot a + y_1 \cdot b + c \\ \vdots &\quad \vdots \quad \vdots \\ X_n &= x_n \cdot a + y_n \cdot b + c \end{aligned} \tag{12.6}$$

mit $n > 3$ Gleichungen für die drei gesuchten Plattenkonstanten a, b und c normalerweise überbestimmt und damit nicht mehr direkt lösbar. Um dieses Problem zu umgehen, bedient man sich der im folgenden beschriebenen Ausgleichsrechnung nach der *Methode der kleinsten Quadrate*.

12.3 Ausgleichsrechnung

Will man allgemein¹ m Unbekannte $\mathbf{x} = (x_1, \dots, x_m)$ aus n Gleichungen

$$\mathbf{a}_i \cdot \mathbf{x} = a_{i1} \cdot x_1 + \dots + a_{im} \cdot x_m = b_i \quad (i = 1, \dots, n) \quad (12.7)$$

mit gegebenen Koeffizienten x_i und a_{ij} bestimmen, dann kann man für $n > m$ nicht erwarten, daß alle Gleichungen gemeinsam erfüllbar sind. Man führt deshalb zunächst als *Residuen* ν_i die Größen

$$\nu_i = a_{i1} \cdot x_1 + \dots + a_{im} \cdot x_m - b_i \quad (i = 1, \dots, n) \quad (12.8)$$

ein. Anstatt nun die Werte x_1, \dots, x_m zu suchen, für die sämtliche ν_i zu Null werden (dann wären alle Gleichungen erfüllt), begnügt man sich damit, die Summe

$$\chi = \sum_{i=1}^n \nu_i^2$$

der Residuenquadrate zu minimieren.

Zur Lösung dieser Aufgabe werden die Gleichungen (12.8) zunächst durch eine Reihe von Umformungen in ein äquivalentes System der Form

$$\begin{aligned} \mu_1 &= r_{11} \cdot x_1 + r_{12} \cdot x_2 + \dots + r_{1m} \cdot x_m - d_1 \\ \mu_2 &= r_{22} \cdot x_2 + \dots + r_{2m} \cdot x_m - d_2 \\ \vdots &= \ddots \quad \vdots \quad \vdots \\ \mu_m &= r_{mm} \cdot x_m - d_m \\ \mu_{m+1} &= -d_{m+1} \\ \vdots &= \vdots \\ \mu_n &= -d_n \end{aligned} \quad (12.9)$$

übergeführt, in dem alle Koeffizienten r_{ij} mit $(i > j)$ gleich Null sind und das demzufolge wesentlich leichter zu behandeln ist.

Sind c und s zwei reelle Zahlen mit $c^2 + s^2 = 1$, dann ändert sich die Summe der Residuenquadrate beispielsweise nicht, wenn man ν_1 durch $\nu'_1 = c \cdot \nu_1 - s \cdot \nu_2$ und ν_2 durch $\nu'_2 = s \cdot \nu_1 + c \cdot \nu_2$ ersetzt, da unter der genannten Voraussetzung

$$\begin{aligned} \nu'_1^2 + \nu'_2^2 &= c^2 \nu_1^2 - 2cs\nu_1\nu_2 + s^2 \nu_2^2 + s^2 \nu_1^2 + 2cs\nu_1\nu_2 + c^2 \nu_2^2 \\ &= (c^2 + s^2) \cdot (\nu_1^2 + \nu_2^2) \end{aligned}$$

gleich $\nu_1^2 + \nu_2^2$ ist. Wählt man speziell

$$\begin{aligned} c &= +a_{11}/h \quad \text{und} \\ s &= +a_{21}/h \quad \text{mit} \quad h = \pm \sqrt{a_{11}^2 + a_{21}^2} \end{aligned}$$

¹Im Interesse einer gebräuchlichen Notation werden einige bisher schon verwendete Symbole zur weiteren Darstellung der Ausgleichsrechnung in einer anderen Bedeutung verwendet. Dies gilt insbesondere für die Größen a , b und x , die nachfolgend nicht die Plattenkonstanten und Sternkoordinaten sondern Koeffizienten und Unbekannte des Gleichungssystems bezeichnen.

dann wird a_{21} wie gewünscht durch

$$a'_{21} = -sa_{11} + ca_{21} = (-a_{21}a_{11} + a_{11}a_{21})/h = 0$$

ersetzt. Durch weitere derartige Umformungen, die auch als Givens-Rotationen bezeichnet werden, lassen sich nacheinander auch die anderen Koeffizienten $a_{31} \dots a_{n1}, a_{32} \dots a_{n2}, \dots, a_{m+1,m} \dots a_{nm}$ eliminieren, bis man schließlich ein Gleichungssystem der Form (12.9) erhält.

Während im ursprünglichen System $\mathbf{Ax} = \mathbf{b}$ die Koeffizientenmatrix $\mathbf{A} = (a_{ij})_{i=1 \dots n, j=1 \dots m}$ im allgemeinen voll besetzt ist, zeichnet sich das transformierte System

$$\begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \mathbf{x} = \mathbf{d} \quad (12.10)$$

durch die Dreiecksgestalt der quadratischen Matrix $\mathbf{R} = (r_{ij})_{i=1 \dots m, j=1 \dots m}$ aus, bei der alle Elemente unterhalb der Diagonalen verschwinden.

Die Summe der Residuenquadrate lässt sich nun in zwei Anteile

$$\chi = |\boldsymbol{\nu}|^2 = |\boldsymbol{\mu}|^2 = \sum_{i=1}^m \mu_i^2 + \sum_{i=m+1}^n d_i^2$$

zerlegen, von denen nur mehr der erste von den Unbekannten x_k abhängt. Die kleinste Summe wird offensichtlich dann erreicht, wenn alle μ_i für $i = 1 \dots m$ verschwinden. Die zugehörigen x_k erhält man durch die Rücksubstitution

$$\begin{aligned} x_m &= d_m / r_{mm} \\ x_k &= (d_k - \sum_{l=k+1}^m r_{kl}x_l) / r_{kk} \quad (k = m-1, \dots, 1) \end{aligned} \quad (12.11)$$

Die Umformung der Gleichungen auf rechte obere Dreiecksgestalt und die anschließende Auflösung sind in der Klasse `SolverLSQ` zusammengefaßt, mit der sich jedes lineare Ausgleichsproblem bequem lösen läßt.

```
// Klasse SolverLSQ zur Lösung von Ausgleichsproblemen
class SolverLSQ
{
public:
    // Konstruktor und Destruktor
    SolverLSQ(int n); ~SolverLSQ();
    // Initialisierung eines neuen Ausgleichsproblems
    void Init();
    // Hinzufügen einer Datengleichung Ax=b zu einem Ausgleichs-
    // problem durch zeilenweise Givens-Rotationen
    void Accumulate (const double A[], double b);
    // Lösung des Ausgleichsproblems durch Rücksubstitution
    void Solve (double x[]);
private:
    int      N;           // Anzahl zu bestimmender Parameter
    double *d;           // Rechte Seite der transformierten Datengleichungen
    double **R;          // Rechte obere Dreiecksmatrix
};
```

Um ein Gleichungssystem $\mathbf{A}\mathbf{x} = \mathbf{b}$ mit n Unbekannten zu behandeln, muß zunächst ein Objekt der Klasse `SolverLSQ` angelegt werden. Anhand der bekannten Dimension wird dabei seitens des Konstruktors dynamischer Speicher für die rechte obere Dreiecksmatrix \mathbf{R} und den Vektor \mathbf{d} des transformierten Systems belegt und initialisiert.

```
// Konstruktor; n: Anzahl der zu bestimmenden Parameter
SolverLSQ::SolverLSQ (int n)
: N(n)
{
    d = new double[N];      // Speicher fuer R und d anfordern
    R = new double*[N];
    for (int i=0; i<N; i++) R[i] = new double[N];
    Init();                // Initialisiere R und d
}
// Initialisierung eines neuen Ausgleichsproblems (Loesche R und d)
void SolverLSQ::Init()
{
    for (int i=0; i<N; i++) { d[i]=0.0; for (int j=0;j<N;j++) R[i][j]=0.0; }
}
```

Danach werden sequentiell für jede Zeile $\mathbf{a}_i \cdot \mathbf{x} = b_i$ ($i = 1, \dots, n$) des Gleichungssystems die Koeffizienten \mathbf{a}_i und b_i mit Hilfe der Methode `Accumulate` prozessiert und in die rechte obere Dreiecksmatrix \mathbf{R} und den Vektor \mathbf{d} transformiert.

```
// Hinzufuegen einer Datengleichung Ax=b zu einem Ausgleichsproblem
// durch zeilenweise Givens-Rotationen
void SolverLSQ::Accumulate (const double A[], double b)
{
    int      i,j;
    double   c,s,h;
    double*  a = new double[N];
    // Kopiere A nach a
    for (i=0;i<N;i++) a[i]=A[i];
    // Konstruktion und Anwendung einer ebenen Givens-Rotation.
    for (i=0; i<N; i++)
    { // Rotation konstruieren und anwenden, um a[i] zu eliminieren.
        if ( R[i][i]==0.0 && a[i]==0.0 ) {
            c = 1.0; s = 0.0; R[i][i] = 0.0;
        }
        else {
            h = sqrt ( R[i][i]*R[i][i] + a[i]*a[i] ); if (R[i][i]<0.0) h=-h;
            c = R[i][i]/h;  s = a[i]/h;  R[i][i] = h;
        };
        a[i] = 0.0;
        // Rotation auf die restlichen Elemente von a anwenden
        for (j=i+1; j<N; j++) {
            h = +c*R[i][j]+s*a[j];  a[j] = -s*R[i][j]+c*a[j];  R[i][j] = h;
        }
        // Rotation auf das i-te Element von d anwenden
        h = +c*d[i]+s*b;  b = -s*d[i]+c*b;  d[i] = h;
    }
    delete[] a;
}
```

Mit Hilfe der Methode `Solve` erhält man anschließend durch Rücksubstitution die gesuchte Lösung des Gleichungssystems.

```
// Loesung des Ausgleichsproblems durch Ruecksubstitution
void SolverLSQ::Solve (double x[])
{
    int i,j; i=j=0;
    double Sum=0.0;
    // Test auf singulaere Matrix
    for (i=0;i<N;i++) {
        if ( R[i][i] == 0.0 ) {
            cerr << " FEHLER: Singulaere Matrix R in SolverLSQ::Solve()" << endl;
            exit(1);
        };
        // Loese Rx=d fuer x_n,...,x_1 durch Rueckwaertseinsetzen
        x[N-1] = d[N-1] / R[N-1][N-1];
        for (i=N-2;i>=0;i--) {
            Sum = 0.0;
            for (j=i+1;j<N;j++) Sum += R[i][j]*x[j];
            x[i] = ( d[i] - Sum ) / R[i][i];
        };
    }
}
```

Alle Feldindizes werden dabei, wie in C++ üblich von 0 bis $n - 1$ gezählt.

12.4 Das Programm FOTO

Das Programm **Foto** ermöglicht eine genaue Positionsbestimmung von Sternen, Kometen oder Kleinplaneten auf Himmelsaufnahmen. Der Anwender wird dabei von der umfangreichen Rechenarbeit entlastet, die die Bestimmung der Standardkoordinaten und die Lösung des Ausgleichsproblems mit sich bringt.

Vor dem Einsatz von **Foto** sind jedoch eine Reihe von Vorbereitungen notwendig, um die verschiedenen Eingabedaten zu erhalten. Mit Hilfe einer Sternkarte identifiziert man zunächst einige hellere Sterne und legt so die ungefähren Koordinaten des Aufnahmegebiets fest. Anschließend wählt man verschiedene Referenzsterne aus, deren äquatoriale Koordinaten in einem Sternkatalog wie dem PPM-Katalog nachgeschlagen werden können. Gegebenenfalls ist dabei die Eigenbewegung der Sterne in der Zeit zwischen der Katalogepoch und dem Aufnahmedatum zu berücksichtigen. Die Vergleichssterne sollten gleichmäßig auf dem Foto verteilt sein und möglichst punktförmig erscheinen, um eine genaue Vermessung zu erlauben. Auf einer Vergrößerung lassen sich die Positionen der Referenzsterne und des untersuchten Objekts problemlos mit transparentem Millimeterpapier bestimmen, das über die Aufnahme gelegt und ungefähr in Nord-Süd-Richtung orientiert wird. Schließlich bestimmt man noch die Rektaszension und die Deklination des Aufnahmefzentrums, die zur Berechnung der Standardkoordinaten benötigt werden. Da Fehler in den Koordinaten der Plattenmitte die Ergebnisse der Positionsbestimmung nur geringfügig beeinflussen, genügt es, diese Werte dem Atlas zu entnehmen.

Sämtliche Daten werden in eine Datei **Foto.dat** eingegeben. Sie enthält in der ersten Zeile die Rektaszension (in $^{\text{h}} \text{ } ^{\text{m}} \text{ }, ^{\text{s}}$) und die Deklination (in $^{\circ} \text{'} \text{''}$) des

Aufnahmzentrums. Anschließend folgen Angaben zu den einzelnen Objekten der Aufnahme. Referenzsterne, die zur Bestimmung der Plattenkonstanten verwendet werden sollen, sind dabei durch einen Stern (*) in der ersten Spalte gekennzeichnet. Nach dem Namen, der bis zu 11 Buchstaben lang sein darf, sind die gemessenen Koordinaten (x, y in mm) und für Referenzsterne zusätzlich die äquatorialen Koordinaten (α in $^{\text{h}}, ^{\text{m}}, ^{\text{s}}$, δ in $^{\circ}, ' , ''$) anzugeben. Die Daten des folgenden Beispiels beziehen sich auf die in Abb. 12.1 skizzierte Aufnahme.

ZENTRUM		12 28 45.000	+44 00 00.00	
*PPM 052974	+47.5	+73.3	12 25 06.952	+44 47 55.79
*PPM 052987	+41.2	-62.1	12 25 59.995	+43 07 05.93
*PPM 052990	+29.9	+65.2	12 26 22.451	+44 42 39.02
*PPM 053019	-4.8	-72.7	12 29 09.766	+43 00 54.07
*PPM 053028	-33.0	+21.0	12 30 52.966	+44 11 19.19
*PPM 053029	-31.2	-62.0	12 30 53.525	+43 09 30.08
*PPM 053050	-51.2	-59.7	12 32 17.476	+43 11 57.76
*PPM 053051	-55.7	+60.6	12 32 21.122	+44 40 54.24
BRADFIELD	+29.2	-42.1		

Die Funktion `GetInput` liest und speichert diese Werte. Anschließend werden die Standardkoordinaten der Referenzsterne bestimmt, die man zur Ausgleichung der Plattenkonstanten benötigt. Mit den Plattenkonstanten lassen sich dann umgekehrt aus den gemessenen Koordinaten die Standardkoordinaten und die äquatorialen Koordinaten aller Objekte berechnen. Für die Referenzsterne erhält man so eine gute Abschätzung des Fehlers, mit dem man bei der Auswertung der Aufnahme rechnen muß. Ferner lässt sich an auffällig großen Fehlern erkennen, ob möglicherweise einzelne Referenzsterne falsch identifiziert oder gemessen wurden.

```
//-----
//  

// Datei: Foto.cpp  

// Zweck: astrometrische Auswertung von Fotoplatten  

//  

// (c) 1999 Oliver Montenbruck, Thomas Pfleger  

//-----  

#include <cmath>  

#include <fstream>  

#include <iomanip>  

#include <iostream>  

#include "APC_Const.h"  

#include "APC_IO.h"  

#include "APC_Math.h"  

#include "APC_Spheric.h"  

using namespace std;  

// Konstanten  

const int MaxObj = 30;      // Maximale Anzahl von Objekten auf der Aufnahme  

const int StrLen = 13;
```

```

//-----
// GetInput: Einlesen von Daten aus einer Datei
//   Filename  Name der Eingabedatei
//   RA0      Rektaszension der Plattenmitte [rad]
//   Dec0     Deklination der Plattenmitte [rad]
//   NObj    Anzahl der eingelesenen Objekte
//   Name[]  Bezeichnungen der Objekte
//   RA[]    Rektaszensionen der Referenzobjekte [rad]
//   Dec[]   Deklinationen der Referenzobjekte [rad]
//   x[]     Gemessene Koordinaten der Referenzobjekte bezogen auf
//   y[]     die Plattenmitte [mm]
//-----

void GetInput ( char* Filename,
                double& RA0, double& Dec0, int& NObj, char Name[][StrLen],
                double RA[], double Dec[], double x[], double y[] )
{
    int      i,k,h,m,deg,min;
    double   s,sec;
    char     c,sign;
    ifstream inp;
    // Eingabedatei zum Lesen oeffnen
    inp.open(Filename);
    cout << " Eingabedatei: " << Filename << endl << endl;

    // Koordinaten der Plattenmitte lesen
    for (k=0;k<StrLen;k++) inp.get(c);
    inp >> h >> m >> s;           inp.get(c).get(c).get(sign);
    inp >> deg >> min >> sec;  inp.ignore(81,'\'\n');
    RA0 = Rad*15.0*Ddd(h,m,s);
    Dec0 = Rad*Ddd(deg,min,sec); if (sign=='-') Dec0 = -Dec0;

    // Namen, Plattenkoordinaten und evtl. aequatoriale Koordinaten lesen
    i = -1;
    do {
        ++i;
        inp.get(Name[i],StrLen);
        if ( Name[i][0] == '*' ) {
            // Referenzstern
            inp >> x[i] >> y[i];
            inp >> h >> m >> s;           inp.get(c).get(c).get(sign);
            inp >> deg >> min >> sec;  inp.ignore(81,'\'\n');
            RA[i] = Rad*15.0*Ddd(h,m,s);
            Dec[i] = Rad*Ddd(deg,min,sec); if (sign=='-') Dec[i] = -Dec[i];
        }
        else {
            // Messpunkt
            inp >> x[i] >> y[i];  inp.ignore(81,'\'\n');
            RA[i]=0.0; Dec[i]=0.0;
        };
        NObj = i+1;
        c=inp.peek(); // Noetig zur Pruefung auf EOF
    }
    while (!inp.eof() && i<MaxObj);
    inp.close(); // Eingabedatei schliessen
}

```

```

//-----
//  

// Hauptprogramm  

//  

//-----  

void main(int argc, char* argv[])
{  

    // Konstanten  

    const int NEst = 3;    // Anzahl der zu schaetzenden Parameter  

    // Variablen  

    int      i, NObj;  

    char     Name[MaxObj][StrLen];  

    double   RAO, Dec0, RA_Obs, Dec_Obs, D_RA, D_Dec;  

    double   RA[MaxObj], Dec[MaxObj], Delta[MaxObj];  

    double   x[MaxObj], y[MaxObj];  

    double   X[MaxObj], Y[MaxObj];  

    double   A[NEst], C[NEst];  

    double   a, b, c, d, e, f;  

    double   Det, FocLen, Scale;  

    SolverLSQ LSQx(NEst);  

    SolverLSQ LSQy(NEst);  

    char     InputFile[APC_MaxFilename] = "";  

    char     OutputFile[APC_MaxFilename] = "";  

    bool    FoundInputfile = false;  

    bool    FoundOutputfile = false;  

    ofstream OutFile;  

    // Titel  

    cout << endl  

        << "      FOTO: astrometrische Auswertung von Himmelsaufnahmen" << endl  

        << "              (c) 1999 Oliver Montenbruck, Thomas Pfleger " << endl  

        << endl;  

    // Ermittle den Namen der Eingabedatei  

    GetFilenames( argc, argv, "Foto.dat", InputFile, FoundInputfile,  

                  OutputFile, FoundOutputfile );  

    // Abbruch, falls die Eingabedatei nicht gefunden wurde  

    if (!FoundInputfile) { cerr << " Abbruch." << endl; exit(-1); }  

    // Eingabedatei lesen  

    GetInput (InputFile, RAO, Dec0, NObj, Name, RA, Dec, x,y);  

    // Standardkoordinaten der Referenzsterne berechnen und  

    // Ausgleichsproblem aufstellen  

    for (i=0;i<NObj;i++)  

        if (Name[i][0]=='*') {  

            EquStd (RA0, Dec0, RA[i], Dec[i], X[i], Y[i]);  

            A[0]=x[i]; A[1]=y[i]; A[2]=1.0;  

            LSQx.Accumulate (A, X[i]);  

            LSQy.Accumulate (A, Y[i]);  

        };  

    // Plattenkonstanten berechnen  

    LSQx.Solve(C); a=C[0]; b=C[1]; c=C[2];  

    LSQy.Solve(C); d=C[0]; e=C[1]; f=C[2];

```

```

// Aequatoriale Koordinaten der Messpunkte (und Fehler fuer die
// Referenzsterne (*) berechnen
for (i=0;i<NObj;i++) {
    X[i] = a*x[i]+b*y[i]+c;
    Y[i] = d*x[i]+e*y[i]+f;
    StdEqu (RA0, Dec0, X[i], Y[i], RA_Obs, Dec_Obs);
    if (Name[i][0]=='*') {
        D_RA = (RA_Obs-RA[i]) * cos(Dec[i]);
        D_Dec = (Dec_Obs-Dec[i]);
        Delta[i] = Arcs * sqrt(D_RA*D_RA+D_Dec*D_Dec); // in []
    };
    RA[i] = RA_Obs; Dec[i] = Dec_Obs;
};

// Brennweite und Abbildungsmassstab berechnen
Det = a*e-d*b;
FocLen = 1.0/sqrt(fabs(Det)); // in [mm]
Scale = Arcs / FocLen; // in ["/mm"]

// Bei Bedarf die Ausgabe in eine Datei umlenken
if (FoundOutputfile) {
    OutFile.open(OutputFile);
    if (OutFile.is_open()) cout=OutFile;
}

// Ausgabe
cout << " Plattenkonstanten:" << endl << endl
    << fixed << setprecision(8)
    << "    a =" << setw(12) << a << "    b =" << setw(12) << b
    << "    c =" << setw(12) << c << endl
    << "    d =" << setw(12) << d << "    e =" << setw(12) << e
    << "    f =" << setw(12) << f << endl << endl;
cout << " effektive Brennweite und Abbildungsmassstab:" << endl << endl
    << fixed << setprecision(2)
    << "    F =" << setw(9) << FocLen << " mm" << endl
    << "    m =" << setw(7) << Scale << " \"/mm" << endl << endl;
cout << " Koordinaten:" << endl << endl
    << "          Name      x      y      X      Y "
    << "          RA       Dec     Fehler" << endl
    << "          mm      mm           "
    << "          h   m   s   o   '   \\"   \" " << endl;
for (i=0;i<NObj;i++) {
    cout << "    " << Name[i] << fixed
        << setprecision(1) << setw(7) << x[i] << setw(7) << y[i]
        << setprecision(4) << setw(9) << X[i] << setw(8) << Y[i]
        << setprecision(2) << setw(14) << Angle(Deg*RA[i]/15,DMMSSs)
        << showpos << " "
        << setprecision(1) << setw(11) << Angle(Deg*Dec[i],DMMSSs)
        << noshowpos;
    if (Name[i][0]=='*') cout << setprecision(1) << setw(6) << Delta[i];
    cout << endl;
}
if (OutFile.is_open()) OutFile.close();
}

```

Mit den Daten des obigen Beispiels liefert Foto die folgende Ausgabe:

FOTO: astrometrische Auswertung von Himmelsaufnahmen
(c) 1999 Oliver Montenbruck, Thomas Pfleger

Eingabedatei: Foto.dat

Plattenkonstanten:

a = 0.00021651 b = 0.00000827 c = 0.00035472
d = -0.00000799 e = 0.00021639 f = -0.00149942

effektive Brennweite und Abbildungsmaßstab:

F = 4616.79 mm
m = 44.68 "/mm

Koordinaten:

Name	x	y	X	Y	RA	Dec	Fehler
	mm	mm			h m s	o ' "	"
*PPM 052974	47.5	73.3	0.0112	0.0140	12 25 07.11	+44 47 50.9	5.2
*PPM 052987	41.2	-62.1	0.0088	-0.0153	12 25 59.96	+43 07 23.9	18.0
*PPM 052990	29.9	65.2	0.0074	0.0124	12 26 22.46	+44 42 25.9	13.2
*PPM 053019	-4.8	-72.7	-0.0013	-0.0172	12 29 09.18	+43 00 54.0	6.4
*PPM 053028	-33.0	21.0	-0.0066	0.0033	12 30 51.88	+44 11 18.0	11.7
*PPM 053029	-31.2	-62.0	-0.0069	-0.0147	12 30 55.30	+43 09 30.5	19.5
*PPM 053050	-51.2	-59.7	-0.0112	-0.0140	12 32 16.68	+43 11 38.5	21.1
*PPM 053051	-55.7	60.6	-0.0112	0.0121	12 32 21.67	+44 41 14.4	21.0
BRADFIELD	29.2	-42.1	0.0063	-0.0108	12 26 45.28	+43 22 39.7	

Nach den Plattenkonstanten sind zunächst die effektive Brennweite (=Kamera-brennweite \times Vergrößerung) und der Abbildungsmaßstab der Aufnahme angegeben. Anschließend folgen für jedes einzelne Objekt die gemessenen rechtwinkeligen Koordinaten (x,y) sowie die daraus berechneten Standardkoordinaten (X,Y) und die äquatorialen Koordinaten Rektaszension und Deklination. Für die Referenzsterne ergeben sich hier typische Abweichungen zu den Katalogörtern von 10" bis 20". Dies entspricht Meßfehlern von rund 1/4–1/2 mm, wie sie bei der Positionsbestimmung mit Millimeterpapier zu erwarten sind. Etwas genauere Ergebnisse lassen sich im allgemeinen mit einem speziellen Meßtisch erreichen.

Die Eigenbewegung im Zeitraum zwischen der Epoche des PPM-Katalogs (2000) und dem Aufnahmezeitpunkt (1982.8) wurde in diesem Beispiel nicht berücksichtigt. Sie beträgt für die ausgewählten Referenzsterne maximal 2" und ist damit wesentlich kleiner als die beobachtete Streuung der Positionsbestimmung.

12.5 Der Position and Proper Motion Sternkatalog

Unter den heute verfügbaren Sternkatalogen bietet der *Position and Proper Motion Catalogue* von S. Röser und U. Bastian hinsichtlich Genauigkeit und Sterndichte eine gute Grundlage für die astrometrische Auswertung von Sternfeldaufnahmen. Er kann überall dort sinnvoll eingesetzt werden, wo Grenzgrößen von heller als

11^m oder Genauigkeiten von rund 1" für die gestellte Aufgabe ausreichen. Lediglich bei höheren Anforderungen muß auf spezielle Kataloge wie den Hipparcos Sternkatalog und den Hubble Guide Star Catalogue zurückgegriffen werden. Mit Hilfe sekundärer Anschlußsterne können dann z.B. auch CCD-Sternfeldaufnahmen mit kleinen Gesichtsfeldern vermessen werden.

Mit Einverständnis der Autoren und des Astronomischen Rechen-Instituts Heidelberg steht der vollständige PPM Katalog einschließlich der zugehörigen Beschreibung auch auf der beiliegenden CD-ROM zur Verfügung. Der Katalog besteht ursprünglich aus vier Teilen, die hier zur einfacheren Handhabung in einer Datei `PPM.dat` zusammengefaßt sind:

- PPM Nord (Num. 1 – 181731)
- PPM Süd (Num. 181732 – 378910)
- PPM Ergänzung um 275 helle Sterne (Num. 400001 – 400321)
- PPM Ergänzung um 90000 südliche Sterne (Num. 700001 - 789676)

Die Datei umfaßt die Daten von insgesamt 468861 Sternen und besteht aus einer entsprechenden Anzahl von Zeilen zu je 131 Zeichen. Die Bedeutung der wichtigsten Angaben ist in Tabelle 12.1 zusammengefaßt.

Tabelle 12.1. Struktur des Position and Proper Motion Sternkatalogs (Auszug)

Spalten	Beschreibung
2– 7	PPM Katalognummer
10– 18	Bonner/Cordoba Durchmusterung Nummer
20– 23	Helligkeit [mag]
25– 26	Spektraltyp
28– 29	Rektaszension J2000 (Stunden)
31– 32	Rektaszension J2000 (Minuten)
34– 39	Rektaszension J2000 (Sekunden)
42–	Deklination J2000 (Vorzeichen)
43– 44	Deklination J2000 (Stunden)
46– 47	Deklination J2000 (Minuten)
49– 53	Deklination J2000 (Sekunden)
56– 62	Eigenbewegung in RA [s/Jahr]
64– 69	Eigenbewegung in Dec ["/Jahr]

Obwohl die Datei nur druckbare ASCII Zeichen enthält und prinzipiell auch mit einem Texteditor bearbeitet werden kann, gestaltet sich das Lesen angesichts einer Größe von ca. 60 MB sehr zeitaufwendig. Auf der CD-ROM sind deshalb zusätzlich die beiden Hilfsprogramme `PPMbin` und `PPMcat` bereitgestellt.

Mit Hilfe von `PPMbin` werden die wichtigsten Informationen (Nummer, Position, Eigenbewegung und Helligkeit) aus dem Katalog ausgelesen und in eine Binärdatei `PPM.bin` umgespeichert:

```
C:\> PPMbin D:\APCd\Data\PPM\PPM.dat PPM.bin
```

```
PPMbin: Konvertierung des PPM Katalogs (ASCII->binaer)
(c) 1999 Oliver Montenbruck, Thomas Pfleger
```

```
*****
```

```
468861 Sterne verarbeitet
```

Diese beansprucht weniger Platz und kann entsprechend schneller verarbeitet werden. Während des Programmelaufs wird für je 10 000 verarbeitete Datensätze ein Sternchen (*) ausgegeben, um auch auf langsameren Rechnern den Arbeitsfortschritt verfolgen zu können. Die Komprimierung nimmt üblicherweise mehrere Minuten in Anspruch. Zur Vereinfachung stehen auf der CD-ROM aber auch fertige Versionen von PPM.bin für Windows und Linux-Betriebssysteme bereit.

Das Programm PPMcat erlaubt es anschließend, alle PPM Sterne innerhalb eines vorgegebenen Gesichtsfeldes aus dem Katalog auszuwählen.

```
>PPMcat D:/APCd/Data/PPM/Win32/PPM.bin
```

```
PPMcat: Auswahl von Sternen aus dem PPM Katalog
(c) 1999 Oliver Montenbruck, Thomas Pfleger
```

Aufnahmezentrum

Rektaszension [hh mm.m]	...	12 28.75
Deklination [Grad]	...	44.0
Radius [Grad]	...	1.0
Epoche [yyyy.y]	...	1982.7

PPM Nummer	RA	Dec	EB(RA)	EB(Dec)	m
	h m s	o ' "	s/Jh	"/Jh	mag
PPM 52981	12 25 45.107	+44 44 03.74	-0.730	-1.90	8.2
PPM 52983	12 25 53.345	+44 37 34.07	+0.120	-3.60	11.6
...					
PPM 53019	12 29 09.759	+43 00 54.09	+0.040	-0.10	9.2
PPM 53025	12 30 06.542	+43 04 12.64	-0.150	+0.50	11.3
PPM 53028	12 30 52.983	+44 11 19.35	-0.100	-0.90	10.2
PPM 53029	12 30 53.483	+43 09 30.24	+0.240	-0.90	9.3
PPM 53039	12 31 39.100	+44 04 05.16	-0.140	-0.30	10.9
PPM 53044	12 31 41.746	+43 13 06.26	-0.440	-1.90	10.6
PPM 53046	12 31 59.476	+43 28 58.62	-0.260	+0.10	10.6
PPM 53051	12 32 21.186	+44 40 54.45	-0.370	-1.20	9.7
PPM 53062	12 33 39.312	+44 05 47.71	-1.300	-6.90	7.9
PPM 53065	12 34 00.512	+44 09 06.19	-0.220	-1.50	10.2
PPM 400176	12 28 04.711	+44 47 39.35	-1.800	-2.00	7.5

Das Ausgabeformat ist so gewählt, daß die Koordinaten einfach vom Programm Foto weiter verarbeitet werden können. Dazu kann beim Start von PPMcat auch der Name einer Ausgabedatei angegeben werden (vgl. Abschn. A.1.3). Der Quellcode von PPMcat ist zusammen mit dem der übrigen Programme auf der CD zum Buch enthalten.

Anhang

A.1 Die beigelegte CD

A.1.1 Inhalt

Die beigelegte CD-ROM enthält die vollständigen Quelltexte aller im Buch beschriebenen Programme sowie die zugehörigen Eingabedateien. Für IBM kompatible PCs sind ferner ausführbare Versionen der Programme beigefügt, die unter den Betriebssystemen Windows 95/98/NT und Linux direkt ausgeführt werden können. Eine Übersicht der einzelnen Verzeichnisse und Dateien gibt die folgende Tabelle. Der Laufwerksbuchstabe des CD Laufwerks (hier D:) kann dabei je nach PC Konfiguration variieren.

Verzeichnis	Beschreibung
D:\APCd\Win32\	Ausführbare Programme (Windows 32-bit Version) AOEcat.exe, Coco.exe, Comet.exe, Eclipse.exe, EclTimer.exe, Foto.exe, Gauss.exe, Luna.exe, Numint.exe, Occult.exe, Phases.exe, Phys.exe, Planpos.exe, Planrise.exe, PPMbin.exe, PPMcat.exe, Sunset.exe
\Linux\	Ausführbare Programme (Linux Version) AOEcat Coco, Comet, Eclipse, EclTimer, Foto, Gauss, Luna, Numint, Occult, Phases, Phys, Planpos, Planrise, PPMbin, PPMcat, Sunset Unix Archive für Daten, ausführbare Programme (nur Linux) und Quelltexte apcddat.tar, apcdexe.tar, apcdsrc.tar,
\Source\	Quelltexte der Bibliotheksmodule (Spezifikations- und Implementierungsdateien) APC_Cheb.h, APC_Cheb.cpp, APC_Const.h, APC_DE.h, APC_DE.cpp, APC_IO.h, APC_IO.cpp, APC_Kepler.h, APC_Kepler.cpp, APC_Math.h, APC_Math.cpp, APC_Moon.h, APC_Moon.cpp, APC_Phys.h, APC_Phys.cpp, APC_Planets.h, APC_Planets.cpp, APC_PrecNut.h, APC_PrecNut.cpp, APC_Spheric.h, APC_Spheric.cpp, APC_Sun.h, APC_Sun.cpp, APC_Time.h, APC_Time.cpp, APC_VecMat3D.h, APC_VecMat3D.cpp GNU_iomanip.h
	Quelltexte der Anwendungsprogramme AOEcat.cpp, Coco.cpp, Comet.cpp, Eclipse.cpp, EclTimer.cpp, Foto.cpp, Gauss.cpp, Luna.cpp, Numint.cpp, Occult.cpp, Phases.cpp, Phys.cpp, Planpos.cpp, Planrise.cpp, PPMbin.cpp, PPMcat.cpp, Sunset.cpp

D:\APCd\Data\

Eingabedateien

Comet.dat, Foto.dat, Gauss.dat, Numint.dat, Occult.dat,
Orkisz.dat, ZC.dat

Asteroid Orbital Elements Datenbank und Dokumentation
\Aoe\astorb.dat, \Aoe\astorb.html

Position and Proper Motion Katalog und Dokumentation
\Ppm\PPM.dat, \Ppm\PPMbss.txt, \Ppm\PPMnorth.txt,
\Ppm\PPMsouth.txt, \Ppm\PPMsupp.txt, \Ppm\ReadMe.txt,
\Ppm\Win32\PPM.bin,

Die Programme \Win32*.exe und die binäre Datei \Data\Ppm\Win32\Ppm.bin sind lediglich zur Ausführung unter Microsoft Windows (32-bit Version für Intel Prozessor) bestimmt. Entsprechendes gilt für die Programme im Verzeichnis \Linux\ und die Datei \Data\Ppm\Linux\PPM.bin, die nur auf Intel x86 PCs unter Linux eingesetzt werden können. Bei den Dateien mit Endungen *.h, *.cpp und *.dat handelt es sich dagegen um ASCII Textdateien, die i.a. auch auf anderen Rechnerarchitekturen und Betriebssystemen (z.B. Unix, Macintosh, VMS) gelesen und verarbeitet werden können. Hierbei werden jedoch unter Umständen die obigen Datei- und Verzeichnisnamen gemäß den jeweiligen Betriebssystemkonventionen umgesetzt.

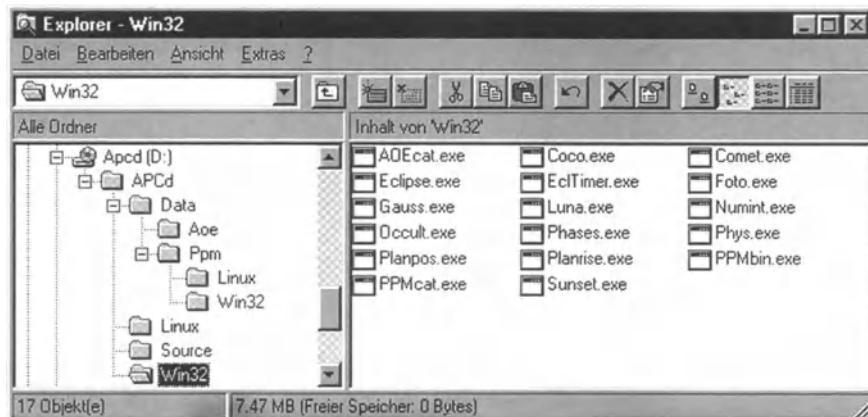


Abb. A.1. Verzeichnisbaum der CD zum Buch in der Darstellung des Windows Explorers

A.1.2 Systemvoraussetzungen

Für die optimale Nutzung der Programme dieses Buches empfehlen wir die Installation auf einem PC mit folgenden Kenndaten:

- Intel Pentium Prozessor 133 MHz oder vergleichbar
- Speicher 16–32 MB
- Freie Festplattenkapazität 10–150 MB

- CD Laufwerk
- Betriebssystem Windows 95/98/NT oder SuSE Linux Version 6.0
- Optional: Microsoft Visual C++ Version 5.0 oder GNU C++ 2.91 (egcs)

Die höheren Anforderungen gelten speziell für Nutzer, die die Quelltexte der Programme selbst übersetzen oder hierauf aufbauend neue Programme entwickeln wollen. Die größeren Anforderungen an Speicherplatz und freiem Festplattenplatz ergeben sich in diesem Fall aus der Nutzung der Entwicklungsumgebung des C++ Compilers und der Anlage temporärer Dateien (z.B. für Klassen- und Browserinformation).

Zur bloßen Nutzung der ausführbaren Programme genügt gegebenenfalls auch ein kleineres PC System (Intel 486, 50 MHz, 8MB). Voraussetzung ist aber in jedem Fall die Installation eines 32-bit Betriebssystems (z.B. Win95 oder Linux). Unter DOS (16-bit) oder Windows 3.1 sind die mitgelieferten Programme (*.exe) nicht lauffähig. Bei geringem Plattenplatz können die Anwendungen i.a. auch direkt von der CD gestartet werden.

Zur Modifikation der Programme und zur Entwicklung eigener Applikationen können prinzipiell verschiedene gängige Compiler und Entwicklungsumgebungen verwendet werden (vgl. Abschn. A.2.2 und A.2.3). Angesichts bestehender Implementierungsunterschiede sind dabei jedoch gegebenenfalls kleinere Anpassungen an den Quelltexten vorzunehmen. Unter Visual C++ 5.0, das bei der Entwicklung unserer Programme eingesetzt wurde, können alle Programme unverändert übersetzt werden.

Neben Windows und Linux können die Quelltexte und Eingabedateien auch auf vielen anderen Rechnern gelesen und verarbeitet werden, soweit diese die ISO 9660 Formatierung (mit Erweiterung) der CD-ROM unterstützen. Angesichts der Vielzahl möglicher Systeme können hierbei jedoch keine allgemeinen und verbindlichen Angaben zu geeigneten Compilern und Entwicklungsumgebungen gemacht werden. Gegebenenfalls sind auch spezielle Anpassungen an den Quelltexten erforderlich.

Zu beachten ist, daß Dateinamen auf der CD von anderen Rechnern möglicherweise nicht korrekt wiedergegeben sondern je nach Betriebssystem verschieden umgesetzt werden (z.B. nur Großbuchstaben). Speziell auf Unix Systemen empfiehlt es sich in diesem Fall, die Quelltexte aus den im Verzeichnis \Linux\ mitgelieferten tar-Archiven zu entpacken, da hierbei die korrekte Schreibweise erhalten bleibt.

A.1.3 Ausführung der Programme

Zur Ausführung der Programme auf einem PC mit Windows Betriebssystem wird das Verzeichnis D:\APCd\Win32 der CD-ROM in ein entsprechendes Verzeichnis (z.B. C:\APCd\Win32) der Festplatte kopiert. In dasselbe Verzeichnis kopiert man anschließend auch die Eingabedateien aus dem Verzeichnis D:\APCd\Data. Nach Start des Kommandointerpreters (DOS Eingabeaufforderung oder command.exe unter Windows 95, cmd.exe unter Windows NT) und Wechsel in das gewählte Programmverzeichnis können die einzelnen Programme wie im folgenden Beispiel

durch Angabe des Programmnamens gestartet werden:

```
Microsoft(R) Windows 95

(C)Copyright Microsoft Corp 1981-1996.

C:>>cd C:\APCd\Exe

C:\APCd\Exe>Sunset

SUNSET: Auf- und Untergangszeiten von Sonne und Mond
(c) 1999 Oliver Montenbruck, Thomas Pfleger

Startdatum (JJJJ MM TT) ...
```

Soweit sinnvoll oder erforderlich, kann bei vielen Programmen die Ein- und/oder Ausgabedatei in der Kommandozeile mit angegeben werden (vgl. Tabelle A.1).

Tabelle A.1. Aufruf der Programme mit optionalen Argumenten

Name	Argumente	Standardeingabe	Standardausgabe
AOEcat	[<in-file> [<out-file>]]	astorb.dat	Bildschirm
Comet	[<in-file> [<out-file>]]	Comet.dat	Bildschirm
Eclipse	[<out-file>]		Bildschirm
Foto	[<in-file> [<out-file>]]	Foto.dat	Bildschirm
Gauss	[<in-file> [<out-file>]]	Gauss.dat	Bildschirm
Numint	[<in-file> [<out-file>]]	Numint.dat	Bildschirm
Occult	[<in-file> [<out-file>]]	Occult.dat	Bildschirm
PPMbin	[<in-file> [<out-file>]]	PPM.dat	PPM.bin
PPMcat	[<in-file> [<out-file>]]	PPM.bin	Bildschirm

Die obigen Hinweise gelten in analoger Weise auch für die Ausführung der Programme unter Linux. Nach Anlegen eines entsprechenden Verzeichnisses und Kopieren der Dateien von der CD können die Programme durch Eingabe des jeweiligen Programmnamens gestartet werden, falls das Arbeitsverzeichnis im Pfad enthalten ist. Wie unter Windows besteht auch unter Linux die Möglichkeit, Ein- und Ausgabedateien in der Kommandozeile anzugeben.

Sollten beim Lesen der CD unter Linux die im Text angegebenen Namen der Programm- und Eingabedateien nicht korrekt wiedergegeben werden, stehen alternativ zwei Unix tar-Archive zur Verfügung. Diese können mit den Befehlern

```
tar -xvf /cdrom/APCd/Linux/apcddat.tar
tar -xvf /cdrom/APCd/Linux/apcdexe.tar
```

entpackt und in das aktuelle Arbeitsverzeichnis kopiert werden.

A.2 Übersetzen und Binden der Programme

Die folgenden Abschnitte beschreiben die notwendigen Schritte zum Übersetzen und Binden der einzelnen Programme und geben Hinweise zu eventuell notwendigen Anpassungen an verschiedene Betriebssysteme und Compiler. Dabei sind aber nur die wichtigsten Probleme angesprochen, die bei der Übertragung der Programme auftreten können. Weitere Hinweise findet man üblicherweise in den Handbüchern des verwendeten Compilers.

A.2.1 Allgemeine Hinweise zur Rechneranpassung

Obwohl Programme, die in der Sprache C++ geschrieben sind, im allgemeinen gut zwischen verschiedenen Rechnern ausgetauscht werden können, gibt es doch einige kleinere Unterschiede zwischen den verschiedenen Implementierungen der Sprache. Schuld daran ist unter anderem die Einführung neuer Sprachelemente (z.B. Namensräume oder Ausnahmebehandlung) und die Erweiterung der C++ Standard Bibliotheken im Rahmen der aktuellen Standardisierungsbestrebungen. Betroffen ist hier von insbesondere die `<iostream>` Bibliothek, deren unterschiedliche Versionen bei mehreren Compilern kleinere Anpassungen der Programme bedingen.

Darüber hinaus können sich Unterschiede aus der Maschinen- und Betriebssystemarchitektur und damit verbundenen, systemspezifischen Wertebereichen der elementaren Ganzzahl- und Fließkommadatentypen ergeben. Wir gehen dabei von einem 32-bit Betriebssystem aus, bei dem 4-Byte int-Zahlen mit einem Wertebereiche von $-2\,147\,483\,648 \dots +2\,147\,483\,647$ und 8-Byte double Zahlen mit einer 15- bis 16-stelligen Genauigkeit verwendet werden. Die meisten Funktionen und Programme arbeiten allerdings auch korrekt, wenn der verwendete Compiler standardmäßig 2-Byte int-Zahlen unterstützt. Lediglich in der Funktion CalDat ist es unbedingt notwendig, anstelle von `int` den Datentyp `long` zu verwenden um einen entsprechend großen Wertebereich sicherzustellen.

Eine Anpassung von Toleranzgrenzen für Iterationen ist im allgemeinen nicht erforderlich, da in den Programmen sinnvolle Werte aus der relativen Maschinengenauigkeit abgeleitet werden. Diese wird im Modul `<limits>` der C++ Standard Bibliothek über die Funktion `numeric_limits<double>::epsilon` bereit gestellt. Alternativ kann man bei älteren C++ Implementierungen das C-Makro `DBL_EPSILON` aus `<float.h>` verwenden, oder mit Hilfe der Funktion

```
//  
// GetAccuracy: Bestimmt die Maschinengenauigkeit  
//  
double GetAccuracy()  
  
    double eps = 1.0e-6;  
    while ( 1.0 + eps > 1.0 )  
        eps *= 0.5;  
    return 2.0*eps;
```

selbst die kleinste `double` Zahl u ermitteln, die bei Addition zu 1.0 einen von 1.0 verschiedenen Wert ergibt. Im Fall der heute allgemein üblichen IEEE Datentypen mit einer Mantisse von 53 Byte beträgt der Wert der relativen Maschinengenauigkeit $u \approx 2.2 \cdot 10^{-16}$.

A.2.2 Microsoft Visual C++ für Windows 95/98/NT

Nachfolgend werden die grundlegenden Arbeitsschritte beschrieben, die zum Erstellen der ausführbaren Programme mit Microsoft Visual C++ 5.0 erforderlich sind. Wir beziehen uns dabei auf die englische Version der Entwicklungsumgebung. Bei Verwendung einer deutschen Version gelten entsprechend abgewandelte Bezeichnungen für Menütitel, Menüeinträge und Schaltflächen, die der jeweiligen Dokumentation der Entwicklungsumgebung zu entnehmen sind. Die verwendeten Projekt- und Verzeichnisnamen stellen eine Empfehlung dar und können bei Bedarf geeignet angepaßt werden.

Zunächst legt man (z.B. mit Hilfe des Windows Explorers) das Unterverzeichnis `C:\APCd\Source\` an und kopiert dorthin alle Dateien aus dem gleichnamigen Verzeichnis der CD zum Buch. Nach Aufruf der Entwicklungsumgebung (MS Developer Studio) legt man im ersten Schritt über den Menüpunkt `File.New.Workspaces` einen neuen Arbeitsbereich (Workspace) an, der zur Verwaltung der einzelnen Programme und Projekte dient. Hier sind dessen Bezeichnung (Vorschlag: `Prj`) und sein Arbeitsverzeichnis (Vorschlag: `C:\APCd\Prj`) anzugeben.

Als erstes Projekt im eben erstellten Workspace wird eine Bibliothek aus den allgemein verwendbaren Modulen erstellt, die zu den Anwendungen hinzugebunden werden kann. Dies vereinfacht die Erstellung der Anwendungen erheblich. Mit `File.New.Projects` wird ein Projekt vom Typ *Win32 Static Library* angelegt, das den Namen `APC_Lib` erhält. Wählen Sie die Option *Add to current workspace* und bestätigen Sie mit *OK* um den Arbeitsbereich für dieses Projekt fertigzustellen.

Anschließend werden die Quelldateien der Bibliotheksmodule zum Projekt `APC_Lib` hinzugefügt. Hierzu ruft man das Menü `Project.Add To Project.Files...` auf und wählt alle Dateien des Ordners `C:\APCd\Source\` aus, auf die das Namensschema `APC_*.cpp` zutrifft. Optional können auch die Headerdateien (`*.h`) mit aufgenommen werden. Dies ist jedoch nicht zwingend erforderlich, da der Compiler das Quellverzeichnis selbstständig nach Headerdateien durchsucht. Nun kann die Bibliothek mit `Build.Build` fertiggestellt werden. Die Entwicklungsumgebung übersetzt dazu zunächst alle Quelltexte und bindet anschließend die erzeugten Objektdateien zu einer statischen Bibliothek `APC_Lib.lib`.

Die Anwendungen `AOEcator` bis `Sunset` werden in jeweils identischer Weise als eigenständige Projekte innerhalb des Workspace `APCd\Prj` angelegt. Im folgenden ist dies am Beispiel von `Coco` beschrieben. Über `File.New.Projects` wählt man ein Projekt vom Typ *Win32 Console Application*. Der Projektname `Coco` wird im entsprechenden Feld eingetragen. Wie bereits beim Erstellen des Projekts `APC_Lib` wählen Sie auch hier die Option *Add to current workspace* und bestätigen mit *OK* um das Projekt zu erstellen. Nun fügen Sie dem Projekt über `Project.Add To Project.Files...` die Datei `Coco.cpp` aus dem Verzeichnis `C:\APCd\Source` hinzu.

und rufen den Menüpunkt **Project Dependencies...** auf. Für das aktuelle Projekt **Coco** markieren Sie nun die Option **APC_Lib**, um der Entwicklungsumgebung mitzuteilen, daß zum Binden von **Coco** die genannte Bibliothek benötigt wird. Das ausführbare Programm **Coco.exe** wird durch Aufruf von **Build.Build** fertiggestellt und kann anschließend über **Build.Execute** gestartet werden.

Für weitere Hinweise verweisen wir auf die umfangreiche Dokumentation des Microsoft C++ Compilers und der Developer Studio Entwicklungsumgebung.

A.2.3 GNU C++ für Linux

Die aktuelle Version 2.91 des GNU C++ Compilers für SuSE Linux 6.0 unterstützt nicht alle **iostream**-Manipulatoren der C++ Standard Bibliothek. Betroffen hiervon sind alle Programme und Module, die bei der Ausgabe die Manipulatoren **right**, **fixed**, **showpos** und **noshowpos** verwenden. Als Abhilfe wurden entsprechende Funktionen

```
#include <iomanip>
#include <iostream>
namespace {
ostream& left (ostream& os){os.setf(ios::left ,ios::adjustfield); return os;};
ostream& right(ostream& os){os.setf(ios::right,ios::adjustfield); return os;};
ostream& fixed(ostream& os){os.setf(ios::fixed,ios::floatfield); return os;};
ostream& showpos (ostream& os){os.setf(ios::showpos); return os;};
ostream& noshowpos(ostream& os){os.unsetf(ios::showpos); return os;};
}
```

in der Datei **GNU_iomanip.h** definiert, die über die Befehle

```
#ifdef __GNUC__ // GNU C++ Anpassungen
#include "GNU_iomanip.h"
#endif
```

in alle Quelltexte mit eingebunden werden kann. Eine weitere Inkompatibilität betrifft die Bestimmung der Maschinengenauigkeit mit Hilfe des Moduls **<limits>** der Standardbibliothek, das zur Zeit unter GNU C++ nicht implementiert ist. Betroffen sind hiervon die Module **APC_DE.cpp** und **APC_Kepler.cpp**, die wie im folgenden Beispiel modifiziert werden können:

```
#ifdef __GNUC__ // GNU C++ Anpassungen
#include <float.h>
#else           // Standard C++ Version
#include <limits>
#endif
...
#endif // GNU C++ Anpassungen
const double umach = DBL_EPSILON;
#else           // Standard C++ Version
const double umach = numeric_limits<double>::epsilon();
#endif
```

Zum Übersetzen und Binden der Programme empfiehlt es sich, zunächst alle Module **APC_*** in einer Bibliothek **libAPC.a** zusammenzufassen:

```
> g++ -c APC*.cpp          # Uebersetzen aller Bibliotheksmodule  
> ar rc libAPC.a APC*.o   # Erzeugung der Bibliothek aus den Objektdateien
```

Unter der Annahme, daß sich alle Dateien im selben Verzeichnis befinden, können die Hauptprogramme anschließend mit den Befehlen

```
> g++ AOEcat.cpp -o AOEcat -lAPC -L. # Uebersetzen und Binden von AOEcat  
> g++ Coco.cpp    -o Coco     -lAPC -L. # Uebersetzen und Binden von Coco  
> ...  
> g++ Sunset.cpp -o Sunset -lAPC -L. # Uebersetzen und Binden von Sunset
```

übersetzt und gebunden werden. Zur einfacheren Anwendung steht auf der CD das Shell-Skript **build** bereit, das die obigen Schritte in einem Arbeitsgang durchführt.

Sollten beim Lesen der CD unter Linux (oder anderen Unixsystemen) die angegebenen Dateinamen nicht korrekt wiedergegeben werden (z.B. wenn der verwendete CD Treiber Erweiterungen des ISO 9660 Formats nicht unterstützt), steht alternativ ein Unix tar-Archiv mit sämtlichen Quelltexten zur Verfügung. Dieses kann mit dem Befehl

```
tar -xvf /cdrom/APCd/Linux/apcdsrc.tar
```

entpackt und in das aktuelle Arbeitsverzeichnis kopiert werden.

A.3 Verzeichnis der Bibliotheksfunktionen

Die APC_Lib Bibliothek umfaßt grundlegende Operatoren, Funktionen und Klassen, die nicht an ein bestimmtes Anwendungsprogramm gebunden sind und sich so besonders für eigene Programmierungen eignen. Insgesamt setzt sich die Bibliothek aus 14 Modulen zusammen, die jeweils einen klar abgegrenzten Aufgabenbereich abdecken:

<code>APC_Cheb.h</code>	Tschebyscheff-Approximation von Funktionen
<code>APC_Const.h</code>	Mathematische und astronomische Konstanten
<code>APC_DE.h</code>	Numerische Integration von Differentialgleichungen
<code>APC_IO.h</code>	Hilfsfunktionen zur Ein- und Ausgabe
<code>APC_Kepler.h</code>	Keplerbahnen
<code>APC_Math.h</code>	Technisch-wissenschaftliche Funktionen
<code>APC_Moon.h</code>	Mondkoordinaten
<code>APC_Phys.h</code>	Physische Ephemeriden
<code>APC_Planets.h</code>	Planetenkoordinaten
<code>APC_PrecNut.h</code>	Präzession und Nutation
<code>APC_Spheric.h</code>	Sphärische Astronomie
<code>APC_Sun.h</code>	Sonnenkoordinaten
<code>APC_Time.h</code>	Zeit- und Kalenderrechnung
<code>APC_VecMat3D.h</code>	Vektor-/Matrixrechnung

Mit Hilfe der genannten Spezifikationsdateien können die jeweils benötigten Module in den Anwendungsprogrammen auf einfache Weise bekannt gemacht werden. Die zugehörigen Implementierungen sind in entsprechenden Dateien `APC_*.cpp` enthalten, die getrennt übersetzt und als Objekt- oder Bibliotheksdateien gebunden werden können. Einzige Ausnahme hiervon ist das Modul `APC_Const`, das keinen Implementierungsteil benötigt und lediglich aus der Spezifikationsdatei besteht.

Die folgende Aufstellung gibt eine Übersicht über alle Bezeichner von öffentlichen Typen, Konstanten, Funktionen und Klassen, die innerhalb der verschiedenen Module bereitgestellt werden.

<code><<</code>	Ausgabeoperator für Datum, Zeit, Winkel, Vektoren und Matrizen
<code>[]</code>	Zugriff auf Vektorkomponenten (kartesisch, polar)
<code>+</code>	Addition von Vektoren und Matrizen
<code>+=</code>	Vektoraddition
<code>-</code>	Subtraktion von Vektoren und Matrizen
<code>-</code>	Unäres Minus (Vektor, Matrix)
<code>-=</code>	Vektorsubtraktion
<code>*</code>	Multiplikation (Skalar, Vektor, Matrix)
<code>/</code>	Division durch einen Skalar
<code>Accumulate</code>	Methode der Klasse <code>SolverLSQ</code> zum Hinzufügen einer Datengleichung
<code>AddThe</code>	Additionstheoreme für Winkelfunktionen

Angle	Hilfsklasse zur Ausgabe von Winkeln
AngleFormat	Aufzählungstyp für Formatangaben zur Winkelausgabe
APC_MaxFilename	Stringlänge für Dateinamen in der Kommandozeile
Arcs	Bogensekunden pro radian
AU	Astronomische Einheit [km]
Bright	Scheinbare Helligkeit eines Planeten
c_light	Lichtgeschwindigkeit [AE/d]
CalDat	Kalenderdatum und Zeit
Cheb3D	Tschebyscheff-Approximation in drei Dimensionen
Col	Spaltenvektoren einer Matrix
Cross	Kreuzprodukt
DateTime	Hilfsklasse zur Ausgabe von Datum und Zeit
Dd	Enumerator vom Typ AngleFormat für dezimale Darstellung
DDd	Enumerator vom Typ TimeFormat für Zeit als Tagesbruchteil
Ddd	Grad-Bruchteile aus Minuten und Sekunden
DE_ACCURACY_NOT_ACHIEVED	Enumerator vom Typ DE_STATE (Flag für zu hohe Genauigkeitsforderung)
DE_DONE	Enumerator vom Typ DE_STATE (Flag für erfolgreichen Integrations schritt)
DE_INIT	Enumerator vom Typ DE_STATE (Neustart des Integrators)
DE_INVALID_PARAMS	Enumerator vom Typ DE_STATE (ungültige Eingaben)
DE_STATE	Aufzählungstyp für Statuscodes des Integrators SolverDE
DE_STIFF	Enumerator vom Typ DE_STATE (Verdacht auf steife Differentialgleichungen)
DE_TOO_MANY_STEPS	Enumerator vom Typ DE_STATE (zu große Schrittzahl)
Deg	$180^\circ/\pi$
Direct	Enumerator vom Typ RotationType (direkte Rotation)
DMM	Enumerator vom Typ AngleFormat für Ausgabe in Grad und ganzen Bogenminuten
DMMm	Enumerator vom Typ AngleFormat für Ausgabe in Grad und dezimalen Bogenminuten
DMMSS	Enumerator vom Typ AngleFormat für Ausgabe in Grad, Bogenminuten und ganzen Bogensekunden
DMMSSs	Enumerator vom Typ AngleFormat für Ausgabe in Grad, Bogenminuten und dezimalen Bogensekunden
DMS	Minuten und Sekunden aus Grad-Bruchteilen
Dot	Skalarprodukt zweier Vektoren
Earth	Enumerator vom Typ PlanetType (Erde)
EccAnom	Exzentrische Anomalie fuer elliptische Bahnen
Ecl2EquMatrix	Transformationsmatrix ekliptikal \rightarrow äquatorial
Elements	Bahnelemente aus Ort und Geschwindigkeit
Elements	Bahnelemente aus zwei Ortsvektoren
Ellip	Ort und Geschwindigkeit für elliptische Bahnen
Equ2EclMatrix	Transformationsmatrix äquatorial \rightarrow ekliptikal

Equ2Hor	Transformation äquatorial → Horizontsystem
EquStd	Standardkoordinaten aus äquatorialen Koordinaten
ETminUT	Approximation der Differenz ET-UT
FileExists	Prüfung, ob eine Datei existiert
FindEta	Sektor-zu-Dreiecks-Verhältnis
Fit	Methode der Klasse Cheb3D zur Tschebyscheff-Approximation einer Funktion
Frac	Nachkommateil einer Zahl
GaussVec	Transformationsmatrix Bahnebene → Ekliptik
GetFilenames	Extraktion von Dateinamen aus der Kommandozeile
GM_Sun	Produkt Gravitationskonstante × Sonnenmasse [AE^3/d^2]
GMST	Mittlere Greenwich-Sternzeit
HHh	Enumerator vom Typ TimeFormat zur Ausgabe von Zeiten in dezimalen Stunden
HHMM	Enumerator vom Typ TimeFormat zur Ausgabe von Zeiten in Stunden und Minuten
HHMMSS	Enumerator vom Typ TimeFormat zur Ausgabe von Zeiten in Stunden, Minuten und Sekunden
Hor2Equ	Transformation Horizontsystem → Äquator
HypAnom	Exzentrische Anomalie für hyperbolische Bahnen
Hyperb	Ort und Geschwindigkeit für hyperbolische Bahnen
Id3D	Einheitsmatrix
Illum	Beleuchtungsparameter der Planeten
index	Aufzählungstyp zur Indizierung der kartesischen Komponenten von Vec3D -Objekten
Init	Methode der Klasse SolverLSQ zur Initialisierung eines Ausgleichsproblems
Integ	Methode der Klasse SolverDE zur Durchführung eines Integrationsschritts
Jupiter	Enumerator vom Typ PlanetType
JupiterPos	Heliozentrische Jupiterkoordinaten
Kepler	Ort und Geschwindigkeit für Keplerbahnen
KepPosition	Planetenposition aus Keplerelementen
KepVelocity	Planetengeschwindigkeit aus Keplerelementen
kGauss	Gauß'sche Gravitationskonstante [AE^3/d^2]
Mars	Enumerator vom Typ PlanetType
MarsPos	Heliozentrische Marskoordinaten
Mat3D	Klasse der 3×3 -Matrizen
Mercury	Enumerator vom Typ PlanetType (Merkur)
MercuryPos	Heliozentrische Merkurkoordinaten
MiniMoon	Mondkoordinaten geringer Genauigkeit
MiniSun	Sonnenkoordinaten geringer Genauigkeit
Mjd	Modifiziertes julianisches Datum
MJD_J2000	Modifiziertes julianisches Datum der Epoche J2000
Modulo	Modulofunktion

MoonEqu	Äquatoriale Mondkoordinaten hoher Genauigkeit
MoonPos	Ekliptikale Mondkoordinaten hoher Genauigkeit
Neptune	Enumerator vom Typ <code>PlanetType</code> (Neptun)
NeptunePos	Heliozentrische Neptunkoordinaten
None	Enumerator vom Typ <code>TimeFormat</code> zur Ausgabe des Datums ohne Uhrzeit
Norm	Betrag eines Vektors
NutMatrix	Nutationsmatrix
Orient	Orientierung des planetozentrischen Bezugssystems
Parab	Ort und Geschwindigkeit für parabolische Bahnen
Pegasus	Nullstellenbestimmung nach dem Pegasus-Verfahren
PertPosition	Planetenpositionen hoher Genauigkeit
phi	Enumerator vom Typ <code>pol_index</code> für Azimut eines Vektors
pi	π
pi2	2π
PlanetType	Aufzählungstyp für Sonne und Planeten
Pluto	Enumerator vom Typ <code>PlanetType</code>
PlutoPos	Heliozentrische Plutokoordinaten
pol_index	Aufzählungstyp für Polarkomponenten von <code>Vec3D</code> -Objekten
Polar	Hilfsklasse zur Definition von Vektoren aus Polarkoordinaten
PosAng	Positionswinkel
PrecMatrix_Ecl	Präzessionsmatrix (ekliptikale Koordinaten)
PrecMatrix_Equ	Präzessionsmatrix (äquatoriale Koordinaten)
Quad	Quadratische Interpolation
r	Enumerator vom Typ <code>pol_index</code> für Betrag eines Vektors
R_Earth	Erdradius [km]
R_Moon	Mondradius [km]
R_Sun	Sonnenradius [km]
R_x	Matrix für elementare Drehung um die x -Achse
R_y	Matrix für elementare Drehung um die y -Achse
R_z	Matrix für elementare Drehung um die z -Achse
Rad	$\pi/180^\circ$
Retrograde	Enumerator vom Typ <code>RotationType</code> (retrograde Rotation)
Rotation	Planetographische Koordinaten
RotationType	Aufzählungstyp zur Bezeichnung der Rotationsrichtung
Row	Zeilenvektoren einer Matrix
Saturn	Enumerator vom Typ <code>PlanetType</code>
SaturnPos	Heliozentrische Saturnkoordinaten
Set	Methode der Klassen <code>Angle</code> , <code>Time</code> und <code>DateTime</code> zur Wahl des Ausgabeformats
Shape	Form und Größe der Planeten
Site	Koordinaten eines Orts an der Erdoberfläche
Solve	Methode der Klasse <code>SolverLSQ</code> zur Rücksubstitution
SolverDE	Klasse zur numerischen Lösung von Differentialgleichungen
SolverLSQ	Klasse zur Lösung von Ausgleichsproblemen

StdEqu	Äquatoriale Koordinaten aus Standardkoordinaten
Stumpff	Stumpff'sche Funktionen
Sun	Enumerator vom Typ PlanetType (Sonne)
SunEqu	Äquatoriale Sonnenkoordinaten
SunPos	Ekliptikale Sonnenkoordinaten
Sys_I	Enumerator vom Typ SystemType (System I)
Sys_II	Enumerator vom Typ SystemType (System II)
Sys_III	Enumerator vom Typ SystemType (System III)
SystemType	Aufzählungstyp zur Bezeichnung des Rotationssystems
T_B1950	Epoche B1950 (in julianischen Jahrhunderten seit J2000)
T_J2000	Epoche J2000 (in julianischen Jahrhunderten seit J2000)
theta	Enumerator vom Typ pol_index für Elevation eines Vektors
Time	Hilfsklasse zur Ausgabe von Zeiten
TimeFormat	Aufzählungstyp für Formatangaben zur Zeit- und Datumsausgabe
Transp	Transponierte Matrix
Uranus	Enumerator vom Typ PlanetType
UranusPos	Heliozentrische Uranuskoordinaten
Value	Methode der Klasse Cheb3D zur Auswertung einer Tschebyscheff-Approximation
Vec3D	Klasse der dreidimensionalen Vektoren
Venus	Enumerator vom Typ PlanetType
VenusPos	Heliozentrische Venuskoordinaten
x	Enumerator vom Typ index für <i>x</i> -Koordinate eines Vektors
y	Enumerator vom Typ index für <i>y</i> -Koordinate eines Vektors
z	Enumerator vom Typ index für <i>z</i> -Koordinate eines Vektors

Bezeichnungen

<i>a</i>	große Halbachse
<i>a</i>	Stationskoeffizient in Länge
<i>A</i>	Azimut
A	Koeffizientenmatrix eines linearen Gleichungssystems
<i>b</i>	ekliptikale Breite
<i>b</i>	Stationskoeffizient in Breite
<i>b</i>	zeitliche Änderung der ekliptikalnen Breite
b	Datenvektor eines linearen Gleichungssystems
<i>B</i>	ekliptikale Breite der Sonne
<i>c</i>	Lichtgeschwindigkeit ($c = 299\,792\,458 \text{ m/s}$)
$c_i(E)$	Stumpffsche Funktionen
<i>C</i>	parallaktischer Winkel
<i>d</i>	Durchmesser des Kernschattens der Erde
d	Richtungsvektor der Rotationsachse
d	Datenvektor eines linearen Gleichungssystems in Dreiecksform
<i>db</i>	Störung der ekliptikalnen Breite
<i>dl</i>	Störung der ekliptikalnen Länge
<i>dr</i>	Störung der Entfernung
<i>D</i>	mittlere Elongation des Mondes von der Sonne
<i>D</i>	Durchmesser des Halbschattens der Erde
<i>e</i>	Exzentrizität
<i>e</i>	Vektor der Länge Eins
<i>E</i>	exzentrische Anomalie
<i>E</i>	Elongation
E	Transformationsmatrix
ET	Ephemeridenzeit
<i>f</i>	Abplattung der Erde oder eines Planeten
<i>f</i>	Öffnungswinkel des Halb-/Kernschattenkegels
<i>f</i>	Koordinatendifferenz in der Fundamentalebene
<i>F</i>	allg.: Anziehungskraft zweier Himmelskörper
<i>F</i>	mittlerer Abstand des Mondes vom aufsteigenden Knoten seiner Bahn
<i>F</i>	Brennweite
<i>g</i>	Koordinatendifferenz in der Fundamentalebene
<i>G</i>	Gravitationskonstante ($G = 2.959122083 \cdot 10^{-4} \text{ AE}^3 M_{\odot}^{-1} d^{-2}$)
GMST	mittlere Sternzeit von Greenwich
<i>h</i>	Höhe über dem Horizont

<i>h</i>	Schrittweite
<i>H</i>	exzentrische Anomalie einer hyperbolischen Bahn
<i>i</i>	Bahnneigung gegen die Ekliptik
<i>i</i>	Phasenwinkel
<i>i</i>	Basisvektor der Fundamentalebene (<i>x</i> -Achse)
<i>j</i>	Basisvektor der Fundamentalebene (<i>y</i> -Achse)
JD	Julianisches Datum
<i>k</i>	Gaußsche Gravitationskonstante ($k = 0.01720209895$)
<i>k</i>	Phase
<i>k</i>	Verhältnis von Mondradius zu Erdradius ($k \approx 0.2725$)
k	Basisvektor senkrecht zur Fundamentalebene (<i>z</i> -Achse)
<i>l</i>	ekliptikale Länge
<i>l</i>	Schattenradius in der Fundamentalebene
<i>l</i>	mittlere Anomalie des Mondes
<i>l'</i>	mittlere Anomalie der Sonne
<i>i</i>	zeitliche Änderung der ekliptikalnen Länge
<i>L</i>	ekliptikale Länge der Sonne
<i>L</i>	Schattenradius am Beobachtungsort
<i>L</i> ₀	mittlere Länge des Mondes
<i>m</i>	scheinbare Helligkeit
<i>m</i>	Abstand des Beobachters von der Schattenachse
<i>M</i>	mittlere Anomalie
<i>M</i>	Größe einer Sonnenfinsternis
<i>M</i> _h	mittlere Anomalie (Hyperbel)
<i>M</i> _○	Masse der Sonne
MJD	Modifiziertes Julianisches Datum
<i>n</i>	Dreiecksflächenverhältnis
<i>n</i>	tägliche Bewegung
N	Nutationsmatrix
<i>p</i>	Bahnpараметer
<i>P</i>	Positionswinkel der Kontaktpunkte einer Sonnenfinsternis
P	Gaußscher Vektor (in Richtung des Perihels)
P	Präzessionsmatrix
<i>q</i>	Periheldistanz
Q	Gaußscher Vektor (senkrecht zum Perihel)
<i>r</i>	allg.: Abstand, Entfernung
<i>r</i>	allg.: Ortsvektor
<i>r̄</i>	Änderung der Entfernung
<i>R</i>	Entfernung Erde-Sonne
<i>R</i>	Radius eines Himmelskörpers
<i>R</i>	Reduktion auf die Ekliptik
<i>R</i>	Refraktion
R	Gaußscher Vektor senkrecht zur Bahnebene
R	Vektor des geozentrischen Sonnenortes
R	Matrix in rechter oberer Dreiecksgestalt

$R_{x,y,z}$	Elementare Drehmatrizen
S	Sektorfläche
t	allg.: Zeit
T	allg.: Zeit in julianischen Jahrhunderten seit J2000
T	Umlaufszeit
T_0	Zeitpunkt des Periheldurchgangs
$T_n(x)$	Tschebyscheff-Polynom n -ten Grades
ΔT	Differenz Ephemeridenzeit - Weltzeit
TDB	Baryzentrische dynamische Zeit
TDT	Terrestrische dynamische Zeit
u	Argument der Breite
U	Hilfsgröße für parabelnahe Bahnen
U	Transformationsmatrix
UT	Weltzeit
UTC	koordinierte Weltzeit
v	Abstand der Spitze des Schattenkegels vom Erdmittelpunkt
v	allg.: Geschwindigkeitsvektor
V	Positionswinkel der Kontaktpunkte einer Sonnenfinsternis
$V(1,0)$	visuelle Einheitshelligkeit
W	Orientierung des Nullmeridians
x, y, z	allg.: kartesische Koordinaten
y	Zustandsvektor
X, Y	Standardkoordinaten (Astrometrie)
z	Hilfswinkel zur Beschreibung der Präzession
α	Rektaszension
α_0	Rektaszension der Rotationsachse
β	ekliptikale Breite
γ	Cosinus der Elongation von der Sonne
$\Delta\beta$	Störung der ekliptikalnen Breite
δ	Deklination
δ_0	Deklination der Rotationsachse
Δ	geozentrische Entfernung
Δ	Dreiecksfläche
ε	Schiefe der Ekliptik
$\Delta\varepsilon$	Nutation in ekliptikaler Breite
ζ	Hilfswinkel zur Beschreibung der Präzession
η	Sektor-zu-Dreieck-Verhältnis
ϑ	allg.: sphärische Koordinate
ϑ	Hilfswinkel zur Beschreibung der Präzession
ϑ	Positionswinkel
Θ	Ortssternzeit
Θ_0	Sternzeit von Greenwich
λ	ekliptikale Länge
λ	geographische Länge (nach Osten positiv)

λ	planetographische Länge (positiv entgegen der Rotation)
λ'	planetozentrische Länge (positiv nach Osten)
$\Delta\lambda$	Störung der ekliptikalnen Länge
Λ	Hilfswinkel zur Beschreibung der Präzession
μ	Hilfsgröße zur Beschreibung der Dreiecksflächenverhältnisse
ν	wahre Anomalie
π	3.1415926...
π	Äquatorial-Horizontalparallaxe
π	Hilfswinkel zur Beschreibung der Präzession
Π	Hilfswinkel zur Beschreibung der Präzession
ϖ	Länge des Perihels
ρ	geozentrische Entfernung
σ	Parameter der Gauß-Lagrangeschen Gleichung
τ	Stundenwinkel
τ	Zwischenzeit
φ	allg.: sphärische Koordinate
φ	geographische (planetographische) Breite
φ'	geozentrische (planetozentrische) Breite
ξ	Residuenquadratsumme
$\Delta\psi$	Nutation in ekliptikaler Länge
ω	Argument des Perihels
Ω	Länge des aufsteigenden Knotens

\emptyset	Durchmesser
Υ	Frühlingspunkt
\oplus	Erde
\odot	Sonne
cy	Jahrhundert
d	Tage
h	Stunden
m	Minuten
m	Größenklassen
r	Umläufe
s	Sekunden
$^\circ$	Grad

Glossar

Aberration: Die Verschiebung des beobachteten Ortes eines Gestirns gegenüber dem geometrischen Ort infolge der endlichen Lichtgeschwindigkeit. (→stellare Aberration; →Lichtlaufzeit).

Äquator: Ein gedachter Großkreis an der Himmelskugel, der auf der Rotationsachse der Erde senkrecht steht. Der Äquator trennt die nördliche und die südliche Himmelshalbkugel von einander und ist gleichzeitig die Bezugsebene für das äquatoriale Koordinatensystem mit den Koordinaten →Rektaszension und →Deklination.

äquatoriale Koordinaten: Auf den →Äquator bezogene Koordinaten (→Rektaszension, →Deklination).

Äquinoktium: Bezugszeitpunkt bei der Angabe astronomischer Koordinaten. Da sich die Lage von Ekliptik und Äquator, die als Bezugsebene für das äquatoriale und das ekliptikale Koordinatensystem dienen, im Lauf der Zeit durch den Einfluß der →Präzession verändert, muß zusätzlich zu jeder Koordinatenangabe auch der Bezugszeitpunkt genannt werden. Nur dann sind die Koordinaten eindeutig festgelegt. Am häufigsten werden das Äquinoktium des Datums, das Äquinoktium →B1950 und das Äquinoktium →J2000 verwendet.

Astrometrie: Die Vermessung von Gestirnpositionen mit Hilfe von Teilkreisen oder fotografischen Aufnahmen.

astrometrische Koordinaten: Koordinaten zum Vergleich mit katalogisierten Sternpositionen. Astrometrische Koordinaten beziehen sich im allgemeinen auf das mittlere →Äquinoktium einer festen Standardepoche (→J2000, →B1950) und berücksichtigen bei Planeten oder Kometen die →Lichtlaufzeit.

Astronomische Einheit: Eine Längeneinheit, die zur Angabe von Entfernungen im Sonnensystem verwendet wird. Eine Astronomische Einheit (AE) ist näherungsweise gleich der mittleren Entfernung zwischen der Erde und der Sonne und beträgt etwa 149.6 Millionen km.

aufsteigender Knoten: Derjenige Bahnpunkt, in dem ein Himmelskörper die Ekliptik von Süden nach Norden durchquert.

Azimut: Eine Koordinate im →Horizontsystem. Das Azimut gibt den von Süden aus nach Westen positiv gezählten Winkel der Richtung an, in der ein Beobachter ein Gestirn sieht.

B1950: Beginn des Besseljahres 1950 ($B1950 = JD\ 2433282.423 = Jan\ 0^d923, 1950$). Lange Zeit gebräuchlicher Bezugspunkt in der astronomischen Zeitzählung, der heute durch die Standardepochen →J2000 abgelöst ist.

Deklination: Der rechtwinklig zum →Äquator gemessene Winkel zwischen einem Gestirn und dem Äquator. Die Deklination bildet zusammen mit der →Rektaszension die Koordinaten des äquatorialen Koordinatensystems. (→Äquinoktium)

Dynamische Zeit (TDB, TDT): Physikalisches Zeitmaß, das zum Beispiel durch Atomuhren realisiert werden kann. Die Dynamische Zeit ist wie die →Ephemeridenzeit eine gleichförmige Zeitzählung, die jedoch dem Zusammenhang von Raum und Zeit in der Relativitätstheorie Rechnung trägt. Während die Baryzentrische Dynamische Zeit (TDB) die Zeit angibt, die ein Beobachter im Schwerpunkt des Sonnensystems messen würde, gibt die Terrestrische Dynamische Zeit (TDT) die Zeit an, die eine Uhr im Erdmittelpunkt anzeigen würde. Beide Zeitdefinitionen unterscheiden sich untereinander und von der →Ephemeridenzeit nur um wenige Millisekunden.

Ekliptik: Derjenige gedachte Großkreis, der als Schnittlinie der Erdbahnebene mit der Himmelskugel entsteht. Von der Erde aus gesehen wandert die Sonne im Lauf eines Jahres einmal durch die Ekliptik. Die Ekliptik dient als Bezugsebene für das ekliptikale Koordinatensystem mit den Koordinaten ekliptikale Länge und ekliptikale Breite (→ekliptikale Koordinaten).

ekliptikale Koordinaten: Der Ort eines Gestirns relativ zur →Ekliptik wird durch die Koordinaten Länge und Breite sowie das dazugehörige →Äquinoxtium festgelegt. Die ekliptikale Länge wird vom →Frühlingspunkt aus längs der Ekliptik nach Osten positiv gezählt. Die ekliptikale Breite ist der senkrecht zur Ekliptik gemessene Winkel zwischen einem Gestirn und der Ekliptik.

Elongation: Der Winkel, unter dem ein Beobachter zwei Gestirne sieht.

Ephemeride: Eine Tabelle, in der die Koordinaten eines Planeten, Kometen oder Asteroiden für einen bestimmten Zeitraum angegeben sind.

Ephemeridenzeit (ET): Eine gleichförmig ablaufende Zeitzählung, die zur Berechnung der Koordinaten eines Planeten, Kometen oder Asteroiden herangezogen wird. Die Ephemeridenzeit wurde eingeführt, um von den unregelmäßigen und nicht vorhersagbaren Schwankungen der Erdrotation, die die Grundlage für die Zählung der →Weltzeit bildet, unabhängig zu sein. Die Differenz zwischen der Weltzeit (UT) und der Ephemeridenzeit (ET) verschwand definitionsgemäß zu Beginn des 20. Jahrhunderts und beträgt gegenwärtig etwa eine Minute.

Frühlingspunkt: Derjenige Schnittpunkt der →Ekliptik mit dem →Äquator, in dem die Sonne auf ihrer scheinbaren jährlichen Bahn den Äquator von Süden nach Norden überschreitet. Dies ist gegenwärtig um den 21. März jeden Jahres der Fall. Der genaue Zeitpunkt dieses Ereignisses definiert den Beginn des Frühlings. Der Frühlingspunkt bildet die Bezugsrichtung zur Zählung der ekliptikalen Länge (→ekliptikale Koordinaten) und der →Rektaszension.

geographische Koordinaten: Zwei Größen (geographische Länge und geographische Breite), die einen Ort auf der Erdoberfläche eindeutig festlegen. Als Bezugsebene der geographischen Koordinaten dient der Erdäquator, während als Ausgangspunkt der Längenzählung nach internationaler Übereinkunft der →Meridian von Greenwich verwendet wird.

geozentrische Koordinaten: Auf den Erdmittelpunkt bezogene Koordinaten.

halber Tagbogen: Die Hälfte desjenigen Winkelbogens einer Gestirnbahn, die über dem →Horizont des Beobachters verläuft. Drückt man den halben Tagbogen im Zeitmaß aus, so gibt er die in →Sternzeit gemessene Zeit vom Aufgang bis zur →Kulmination oder von der Kulmination bis zum Untergang des Gestirns an.

heliographische Koordinaten: Analog zu den →geographischen und planetographischen Koordinaten definierte Größen zur eindeutigen Festlegung von Orten an der Oberfläche der Sonne.

heliozentrische Koordinaten: Auf den Mittelpunkt der Sonne bezogene Koordinaten.

Höhe: Darunter versteht man in der sphärischen Astronomie den von einem Beobachter auf der Erdoberfläche gemessenen Winkel zwischen einem Gestirn und dem →Horizont. Die tatsächliche Höhe eines Gestirns weicht infolge der →Refraktion von seiner beobachteten Höhe um bis zu 35' ab.

Horizont: Die gedachte Schnittlinie einer Tangentialebene an die Erdoberfläche im Standpunkt des Beobachters mit der Himmelskugel. Der Horizont dient als Bezugsebene für das →Horizontsystem mit den Koordinaten →Azimut und →Höhe.

Horizontsystem: Ein Koordinatensystem, das sich auf den lokalen Horizont des Beobachters bezieht und in dem das →Azimut und die →Höhe als Koordinaten verwendet werden. Das Horizontsystem findet beispielsweise bei der Berechnung von Auf- und Untergangszeiten der Gestirne Verwendung. Zur Angabe von Ephemeriden in einem Jahrbuch ist es jedoch ungeeignet, weil Azimut und Höhe eines Gestirns vom Beobachtungsort abhängen und sich infolge der Erddrehung ständig verändern.

J2000: Standardepoche, auf die in der astronomischen Zeitzählung immer wieder Bezug genommen wird. Sie fällt mit dem Mittag des ersten Januars 2000 (1.5 Jan 2000 = JD 2451545.0)

zusammen. Der Buchstabe „J“ kennzeichnet, daß es sich um eine julianische Standardepoche handelt. Aufeinanderfolgende julianische Standardepochen unterscheiden sich üblicherweise um volle julianische Jahrhunderte zu je 36525 Tagen (z.B. J1900 = JD 2415020.0). (→B1950)

Keplerproblem: Andere Bezeichnung für das →Zweikörperproblem.

Kulmination: Der Moment, in dem ein Gestirn seine größte (oder kleinste) Höhe über dem →Horizont erreicht. Die Kulmination findet statt, wenn das Gestirn den →Meridian passiert.

Lichtlaufzeit: Während der Zeit, die das Licht benötigt, um die Strecke zur Erde zurückzulegen (499 Sekunden pro Astronomische Einheit), bewegt sich ein Planet in seiner Bahn ein kleines Stück weiter. Der beobachtete Ort entspricht deshalb nicht dem tatsächlichen Ort des Planeten zur Beobachtungszeit.

Mehrkörperproblem: Die Aufgabe, die Bewegung von mehr als zwei Körpern unter dem Einfluß ihrer gegenseitigen Anziehungskräfte zu berechnen. Während sich das Zweikörperproblem geschlossen lösen läßt, ist dies beim Mehrkörperproblem im allgemeinen nicht möglich. Zur mathematischen Behandlung verwendet man neben analytischen Näherungen (Reihenentwicklungen) meist numerische Methoden.

Meridian: In der Astronomie bezeichnet man so einen Großkreis an der Himmelskugel, der durch den →Zenit des Beobachters läuft und den →Horizont im Nordpunkt und im Südpunkt schneidet (→Kulmination). In der Geographie bezeichnet der Begriff „Meridian“ einen Großkreis auf der Erdoberfläche, in dessen Ebene die Erdachse liegt. Alle Meridiane verlaufen durch die Erdpole und schneiden den Erdäquator unter einem rechten Winkel. Meridiane dienen zur Angabe der geographischen Länge eines Ortes auf der Erde.

Nullmeridian: Derjenige Längengrad im System →planetographischer Koordinaten, der als Referenzmeridian der Längenzählung definiert wurde.

Nutation: Eine der →Präzession überlagerte Schwingung der Erdachse um ihre mittlere Lage. Die Periode der Nutation beträgt 18.6 Jahre und wird durch den Umlauf des aufsteigenden Knotens der Mondbahn bestimmt.

Parallaxe: Durch Variation des Beobachtungsortes hervorgerufene scheinbare Positionsveränderung eines Gestirns vor dem Hintergrund der Fixsterne. Die Parallaxe ist für Gestirne in geringer Entfernung am größten und macht sich daher hauptsächlich beim Mond und (in weitaus geringerem Maße) bei erdnahen Kometen und Asteroiden bemerkbar (→topozentrische Koordinaten).

perifokale Koordinaten: Position eines Himmelskörpers relativ zur Bahnebene und zur Apсидenlinie.

Perigäum: Der erdnächste Punkt der Mondbahn oder einer Satellitenbahn.

Perihel: Der sonnennächste Punkt einer Planeten- oder Kometenbahn.

physische Ephemeriden: Diejenigen Größen, die den Anblick eines Himmelskörpers im Teleskop beschreiben. Hierzu zählen z.B. der scheinbare Durchmesser, die Helligkeit, die Lage des →Zentralmeridians und der →Positionswinkel der Rotationsachse.

planetographische Koordinaten: Winkelgrößen zur Festlegung von Orten an der Oberfläche eines Himmelskörpers, die den geographischen Koordinaten auf der Erde entsprechen.

Positionswinkel: Ein an der scheinbaren Himmelssphäre gemessener Winkel zwischen einer Bezugsrichtung und einer gegebenen Blickrichtung (z.B. zur Sonne oder zum Pol eines Planeten). Üblicherweise wird der Positionswinkel von Norden ausgehend über Osten zunehmend im Gradmaß gezählt.

Präzession: Die langfristige Verlagerung der Ekliptik und des Himmelsäquators. Durch die Störkräfte der Sonne und des Mondes steht die Rotationsachse der Erde nicht fest im Raum. Sie wandert mit einer Periode von 26000 Jahren auf einem Kegelmantel um den mittleren Pol der Ekliptik. Dementsprechend verändert sich auch die Orientierung des Himmelsäquators. Die Kräfte der Planeten auf die Erdbahnebene führen zusätzlich zu einer langsamen Verlagerung

der Ekliptik. Die Wanderung von Äquator, Ekliptik und →Frühlingspunkt erfordert, daß bei den →äquatorialen und →ekliptikal Koordinaten eines Gestirns immer ein Bezugszeitpunkt (→Äquinoktium) angegeben werden muß.

Refraktion: Die Brechung von Lichtstrahlen in der Erdatmosphäre. Einem Beobachter am Boden erscheint ein Gestirn in einer etwas größeren Horizonthöhe, als ohne den Einfluß der Refraktion. In Horizontnähe muß ein Lichtstrahl einen besonders langen Weg durch die Atmosphäre zurücklegen. Daher macht sich die Refraktion beim Auf- oder Untergang der Gestirne am stärksten bemerkbar.

Rektaszension: Eine Koordinate des →äquatorialen Koordinatensystems. Die Rektaszension wird vom →Frühlingspunkt aus längs des →Äquators nach Osten positiv gezählt. Der sich ergebende Winkel wird üblicherweise im Zeitmaß ($15^\circ \equiv 1^h$) mit einer Unterteilung in Stunden, Minuten und Sekunden angegeben.

scheinbare Koordinaten: Koordinaten eines Gestirns, wie sie zum Beispiel beim Einstellen mit den Teilkreisen eines Fernrohrs benötigt werden. Scheinbare Koordinaten beziehen sich auf die aktuelle Orientierung der Erdachse und beinhalten deshalb Korrekturen für →Präzession und →Nutation. Berücksichtigt wird ferner die →stellare Aberration und bei Körpern des Sonnensystems die →Lichtlaufzeit.

stellare Aberration: Der Unterschied zwischen der Richtung eines Lichtstrahls für einen relativ zur Sonne ruhenden und einen sich mit der Erde mitbewegenden Beobachter.

Sternzeit: Die Rektaszension der Sterne, die ein Beobachter gerade im Meridian sieht. Ein Sterntag entspricht der Dauer einer Erdumdrehung und dauert rund $23^h 56^m$ Weltzeit.

Stundenwinkel: Differenz zwischen der lokalen →Sternzeit und der →Rektaszension eines Gestirns. Der Stundenwinkel ist damit ein Maß für die Zeit, die seit der letzten →Kulmination des Himmelskörpers vergangen ist.

topozentrische Koordinaten: Koordinaten, die sich auf den Ort des Beobachters auf der Erdoberfläche beziehen. Topozentrische und →geozentrische Koordinaten unterscheiden sich durch die →Parallaxe.

Universal Time (UT): Englische Bezeichnung für die →Weltzeit.

Weltzeit: Eine aus der Erddrehung abgeleitete Zeitzählung, die die Grundlage der bürgerlichen Zeitrechnung bildet. Die Weltzeit kann aus der Beobachtung des Meridiandurchgangs von Gestirnen bekannter →Rektaszension ermittelt werden. Da die Erdrotation ungleichförmigen Schwankungen unterworfen ist, bildet die Weltzeit keine kontinuierlich verlaufende Zeitzählung (→Ephemeridenzeit, →Dynamische Zeit).

Zenit: Der senkrecht über einem Beobachter gedachte Punkt der Himmelskugel. Die →Höhe des Zenits beträgt also 90° .

Zentralmeridian: Der durch den Mittelpunkt des scheinabren Planetenscheibchens verlaufende Längengrad. Entspricht der planetographischen Länge der Erde.

Zweikörperproblem: Die Aufgabe, die Bewegung zweier Körper unter dem Einfluß ihrer gegenseitigen Anziehungskräfte zu berechnen. Das Zweikörperproblem beschreibt in vereinfachender Weise die Bewegung eines Planeten, Kometen oder Asteroiden um die Sonne, wobei Störungen durch die anderen Körper im Sonnensystem vernäglässigt werden. Die Bedeutung des Zweikörperproblems für die Ephemeridenrechnung liegt in seiner relativ einfachen mathematischen Lösbarkeit begründet.

Literaturverzeichnis

Die folgende Aufstellung enthält eine Auswahl grundlegender und weiterführender Werke aus dem Bereich der sphärischen Astronomie und Himmelsmechanik sowie verschiedene Titel zur numerischen Mathematik und zur Programmierung. Sie soll dem Leser die Möglichkeit zur Vertiefung einzelner Themen geben, die hier nur knapp behandelt werden konnten. Soweit sinnvoll sind neben Literaturstellen auch ergänzende Verweise auf World-Wide-Web Seiten aufgeführt. Naturgemäß unterliegen Internet-Adressen allerdings einem häufigen Wechsel. Sollte eine der genannten Angaben nicht mehr aktuell sein, führt meist die Suche mit einer der gängigen Internet-Suchmaschinen (Altavista, Yahoo, etc.) zum Ziel.

Allgemeine Werke

- [1] *Astronomical Almanac*; U. S. Government Printing Office, Her Majesty's Stationery Office; Washington, London.
Führendes Jahrbuch im englischsprachigen Raum. Die Ausgabe von 1982 enthält eine Zusammenstellung der Zahlenwerte und Konstanten (Sternzeit, Präzession und Nutation, Planetenmassen- und -durchmesser, Elemente der physischen Ephemeriden etc.), die derzeit von der IAU für astronomische Berechnungen empfohlen werden.
- [2] H. Bucerius, M. Schneider; *Himmelsmechanik I-II*; Bd. 143/144, Bibliographisches Institut; Mannheim (1966).
Moderne Darstellung der Himmelsmechanik. Neben anderen Themen werden die allgemeine Störungstheorie, die Theorie der Mondbahn und die Bahnbestimmung behandelt.
- [3] L. E. Doggett, G. H. Kaplan, P. K. Seidelmann; *Almanac for Computers for the Year 19xx*; Nautical Almanac Office, United States Naval Observatory; Washington.
Diese jährliche Veröffentlichung enthält Darstellungen der Koordinaten von Sonne, Mond und Planeten durch Tschebyscheff-Polynome, die eine einfache und genaue Berechnung beliebiger Ephemeriden auch auf kleinen Rechnern erlauben. Die einleitenden Kapitel geben viele Hinweise und einfache Formeln zur Berechnung astronomischer Vorgänge.
- [4] *Explanatory Supplement to the American Ephemeris and Nautical Almanac*; U. S. Government Printing Office, Her Majesty's Stationery Office; Washington, London (1974); Vollständig überarbeitete Neuauflage von P. K. Seidelmann (ed.); University Science Books (1992).
Ergänzungsband zu den jährlichen Ausgaben des *Astronomical Almanac*. Dokumentation zu den verwendeten Daten und Rechenverfahren.
- [5] R. M. Green; *Spherical Astronomy*; Cambridge University Press; Cambridge (1985).
Modernes Standardwerk zur sphärischen Astronomie.
- [6] A. Guthmann; *Einführung in die Himmelsmechanik und Ephemeridenrechnung*; Spektrum Akademie Verlag (ehem. BI Wissenschaftsverlag), Heidelberg (1994).
Verständliche und praxisorientierte Darstellung der Himmelsmechanik.
- [7] D. McNally; *Positional Astronomy*; Muller Educational; London (1974).
Einführung in die sphärische Astronomie.

- [8] J. Meeus; *Astronomical Algorithms*; Willmann-Bell; Richmond, Virginia (1991). Überarbeitete und wesentlich erweiterte Neuauflage des Buches *Astronomical Formulae for Calculators*. Praxisbezogene Sammlung interessanter und ausgefälliger Rechenverfahren für Amateurastronomen. Ausführliche Beispiele, die auch auf kleinen Rechnern nachvollzogen werden können. Zum Buch sind Programmdisketten in Basic, Pascal und C erhältlich.
- [9] J. Meeus; *Astronomical Tables of the Sun, Moon and Planets*; Willmann-Bell; Richmond, Virginia (1983). Umfangreiche Zusammenstellung interessanter astronomischer Daten und Ereignisse wie Mondphasen, Sonnen- und Mondfinsternisse, Planetenkonstellationen, Bedeckungen von Planeten durch den Mond.
- [10] I. I. Mueller; *Spherical and practical astronomy*; Frederick Ungar Publishing Co.; New York (1969). Darstellung der sphärischen Astronomie und ihrer Anwendung in der Geodäsie. Behandelt werden unter anderem die Grundlagen und die praktische Realisierung der verschiedenen Zeitdefinitionen sowie die Vorhersage und Auswertung von Sonnenfinsternissen und Sternbedeckungen.
- [11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery; *Numerical Recipes*; Cambridge University Press; Cambridge, 2. Aufl. (1992). Umfassende Sammlung wichtiger Verfahren aus allen Gebieten der numerischen Mathematik. Erhältlich als Fortran-, Pascal- oder C-Version. Die Programme sind zusätzlich auf Diskette verfügbar.
- [12] A. E. Roy; *Orbital Motion*; Adam Hilger Ltd.; Bristol, 2nd ed. (1982). Allgemeine Einführung in die Himmelsmechanik.
- [13] H. R. Schwarz; *Numerische Mathematik*; B. G. Teubner Verlag; Stuttgart, 2. Aufl. (1988). Praxisbezogene Darstellung wichtiger Algorithmen der numerischen Mathematik.
- [14] K. Stumpff; *Himmelsmechanik I-III*; VEB Deutscher Verlag der Wissenschaften; Berlin (1959, 1965, 1974). Gesamtdarstellung der Himmelsmechanik.
- [15] B. Stroustrup; *Die C++ Programmiersprache*; Addison Wesley; Bonn; 3. Auflage (1998). Umfassendes Lehrbuch vom Entwickler der Sprache C++ unter Berücksichtigung des aktuellen Sprachstandards und der Standard Template Library.

Zu Kapitel 2 (Koordinatensysteme)

- [16] J. H. Lieske, T. Lederle, W. Fricke, B. Morando; *Expressions for the Precession Quantities Based upon the IAU (1976) System of Astronomical Constants*; Astronomy and Astrophysics, vol. 58, pp. 1-16 (1977). Aufstellung der verschiedenen Hilfswinkel, die zur Beschreibung der Präzession in ekliptikalnen und äquatorialen Koordinaten benötigt werden. Die numerischen Werte basieren auf dem aktuellen System astronomischer Konstanten der IAU von 1976.
- [17] G. Moyer; *The Origin of the Julian Day System*; Sky and Telescope, vol. 61, pp. 311-313 (April 1982). Aufsatz zur Geschichte des Julianischen Datums mit Formeln zur gegenseitigen Umrechnung zwischen der fortlaufenden Tageszählung und dem gewöhnlichen Kalender.

Zu Kapitel 3 (Auf- und Untergänge)

- [18] L. D. Schmadel, G. Zech; *Empirical transformations from UT to ET for the period 1800-1988*; Astronomische Nachrichten, vol. 309, pp. 219-221 (1988).
 Darstellung der beobachteten Differenz von Weltzeit und Ephemeridenzeit durch ein Polynom zwölften Grades mit einem Gültigkeitsbereich von fast zweihundert Jahren.

Algorithmen zur Berechnung von Auf- und Untergängen und zur Behandlung der Refraktion sind im *Almanac for Computers* [3] angegeben.

Zu Kapitel 4 (Kometenbahnen)

- [19] J. M. A. Danby, T. M. Burkhardt; *The Solution of Kepler's Equation I-II*; Celestial Mechanics, vol. 31, pp. 95-107, pp. 317-328 (1983).
 Diskussion verschiedener Iterationsverfahren zur numerischen Behandlung der Keplergleichung und der zugehörigen Startwerte.
- [20] B. Marsden; *Catalogue of Cometary Orbits*; Smithsonian Astrophysical Observatory; Cambridge, Mass.
 Regelmäßig überarbeitete Zusammenstellung oskulierender Bahnelemente der bekannten Kometen.
- [21] *Ephemerides of Minor Planets*; Institut für Theoretische Astronomie; St. Petersburg.
 Bahnelemente und Ephemeriden der bekannten Kleinplaneten. Erscheint jährlich.
- [22] Yu. V. Batrakov, V. A. Shor; *Catalogue of Orbital elements and Photometric parameters of 7316 Minor Planets numbered by 25 November, 1996*; Institut für Theoretische Astronomie; St. Petersburg, Russia (1997).
 Aktuelle Bahnelemente der wichtigsten katalogisierten Kleinplaneten. Erhältlich beim Astronomical Data Center der NASA unter
<http://adc.gsfc.nasa.gov/adc-cgi/cat.pl?/catalogs/1/1245/>

Die Behandlung parabelnaher Bahnen nach Stumpff ist in *Himmelsmechanik I* [14] beschrieben. Eine umfangreiche Datenbank mit Kleinplanetenbahnelementen wird vom Lowell Observatorium unter <ftp://ftp.lowell.edu/pub/elgb/astorb.html> bereitgestellt.

Zu Kapitel 5 (Störungsrechnung)

- [23] L. F. Shampine, M. K. Gordon; *Computer Solution of ordinary Differential Equations*; Freeman and Comp., San Francisco (1975);
 deutsch unter dem Titel *Computer-Lösung gewöhnlicher Differentialgleichungen*; Vieweg Verlag, Braunschweig (1984).
 Einführung in die Theorie der Mehrschrittverfahren zur numerischen Lösung gewöhnlicher Differentialgleichungen und Beschreibung des Adams-Verfahrens variabler Ordnung und Schrittweite DE.
- [24] X. X. Newhall, E. M. Standish Jr., J. G. Williams; *DE102: a numerically integrated ephemeris of the moon and planets spanning fourty-four centuries*; Astronomy and Astrophysics, vol. 125, pp. 150-167 (1983);
 Beschreibung der seinerzeit besten Ephemeride über einen Zeitraum von mehreren tausend Jahren. Die beschriebenen Verfahren gelten im wesentlichen auch für alle weiteren Development Ephemeriden des JPL. Aktuelle Versionen (DE200, DE405, etc.) sind erhältlich unter
http://ssd.jpl.nasa.gov/eph_info.html

Aktuelle Bahnelemente für Kleinplaneten werden jährlich in den *Ephemerides of Minor Planets* ([21], siehe auch [22]) des St. Petersburger Instituts für Theoretische Astronomie sowie verschiedenen astronomischen Jahrbüchern veröffentlicht.

Zu Kapitel 6 (Planetenebahnen)

- [25] P. Bretagnon, J.-L. Simon; *Tables for the motion of the sun and the five bright planets from -4000 to +2800; Tables for the motion of Uranus and Neptun from +1600 to +2800;* Willmann-Bell; Richmond, Virginia (1986).
 Reihenentwicklungen und Programme, mit denen die Bewegung der inneren Planeten auch in historischen Zeiträumen mit ausreichender Genauigkeit berechnet werden kann. Die Koordinaten der äußeren Planeten werden abschnittsweise durch Polynome dargestellt.
- [26] T. C. van Flandern, K. F. Pulkkinen; *Low precision formulae for planetary positions;* Astrophysical Journal Supplement Series, vol. 41, p. 391; (1979).
 Reihenentwicklungen zur Berechnung heliozentrischer und geozentrischer Planetenkoordinaten mit niedriger Genauigkeit (ca. 1').
- [27] E. Goffin, J. Meeus, C. Steyart; *An accurate representation of the motion of Pluto;* Astronomy and Astrophysics, vol. 155, p. 323; (1986).
 Darstellung der Plutobahn durch eine trigonometrische Reihe, die durch Anpassung an eine numerisch integrierte Ephemeride gewonnen wurde.
- [28] G. W. Hill; *Tables of Jupiter, Tables of Saturn;* Astronomical Papers of the American Ephemeris, vol. VII, part 1-2; Washington (1898).
 Analytische Reihenentwicklungen der Bewegung von Jupiter und Saturn.
- [29] M. P. Jarnagin, jr.; *Expansions in elliptic motion;* Astronomical Papers of the American Ephemeris, vol. XVIII; Washington(1965).
 Reihenentwicklung der Mittelpunktsgleichung und des Radius sowie weiterer Koordinaten im ungestörten Keplerproblem bis zu hohen Ordnungen in der Exzentrizität.
- [30] S. Newcomb; *Tables of the motion of the Earth, Tables of the heliocentric motion of Mercury, Tables of the heliocentric motion of Venus, Tables of the heliocentric motion of Mars;* Astronomical Papers of the American Ephemeris, vol. VI, part 1-4; Washington (1898).
 Analytische Reihenentwicklungen zur Bewegung der inneren Planeten.
- [31] S. Newcomb; *Tables of the heliocentric motion of Uranus, Tables of the heliocentric motion of Neptune;* Astronomical Papers of the American Ephemeris, vol. VII, part 3-4; Washington (1898).
 Reihenentwicklung der Uranus- und Neptunkoordinaten.
- [32] F. E. Ross; *New Elements of Mars;* Astronomical Papers of the American Ephemeris, vol. IX, part 2; Washington (1917).
 Verbesserung der Bahnelemente aus Newcombs Tafeln der Marsbahn.

Zu Kapitel 7 (Physische Ephemeriden)

- [33] M. E. Davies, V. K. Abalakin, M. Bursa, J. H. Lieske, B. Morando, D. Morrison, P. K. Seidelmann, A. T. Sinclair, B. Yallop, Y. S. Tjuflin; *Report of the IAU/IAG/COSPAR Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 1994;* Celestial Mechanics and Dynamical Astronomy vol. 63, 127-148 (1996).
 Beschreibt die Rotationselemente der großen Planeten, die derzeit im *Astronomical Almanac* [1] verwendet werden. Die aktuellen Ergebnisse der Arbeitsgruppe für kartographische Koordinaten der Astronomischen Gesellschaft werden etwa alle drei Jahre veröffentlicht (*Celestial Mechanics* vol. 22, pp. 205-230 (1980), vol. 29, pp. 309-321 (1983), vol. 39, pp. 103-113 (1986), vol. 46, pp. 187-204 (1989), vol. 53, pp. 377-397 (1992)).

Ausführliche Hinweise zur Berechnung physischer Ephemeriden gibt außerdem das *Explanatory Supplement* [4].

Zu Kapitel 8 (Mondbahn)

- [34] E. W. Brown; *An introductory treatise on the Lunar Theory*; Cambridge University Press (1896), Dover Publications (1960).
Darstellung der verschiedenen störungstheoretischen Methoden zur analytischen Behandlung der Mondbewegung.
- [35] M. Chapront-Touzé, J. Chapront; *ELP 2000-85: a semi-analytical lunar ephemeris adequate for historical times*; Astronomy & Astrophysics, vol. 190, 342 (1988).
Mittlere Bahnelemente und Störungsterme zur Darstellung der Mondbahn über lange Zeiträume.
- [36] M. C. Gutzwiller, D. S. Schmidt; *The motion of the moon as computed by the method of Hill, Brown and Eckert*; Astronomical Papers of the American Ephemeris, vol. XXIII, part 1; Washington (1986).
Analytische Reihenentwicklung des Hauptproblems der Mondbewegung (ohne Berücksichtigung planetarer Störungen).
- [37] *Improved Lunar Ephemeris 1952-1959*; Nautical Almanac Office; Washington, 1954.
Überarbeitete Version der Brownschen Mondtheorie von 1954 mit allen benötigten Störungstermen und den mittleren Argumenten.

Eine gute Einführung in die Theorie der Mondbahn und die grundlegenden Störungen findet man auch in dem Band *Himmelsmechanik II* von Bucerius und Schneider [2]. Die Approximation von Funktionen durch Tschebyscheff-Polynome ist in den Lehrbüchern zur numerischen Mathematik erläutert (siehe [11] und [13]).

Zu Kapitel 9 (Sonnenfinsternisse)

- [38] F. Espanak; *Fifty Year Canon of Solar Eclipses*; NASA Reference Publication 1178 Revised; NASA (1987).
Elemente und Karten der Sonnenfinsternisse im 20. und 21. Jahrhundert mit detaillierten Angaben zu den Finsternissen von 1986 – 2035.
- [39] H. Mucke, J. Meeus; *Canon of Solar Eclipses, -2003 to 2526*; Astronomisches Büro, Hasenwirtgasse 32, Wien (1984).
Elemente von 10774 Sonnenfinsternissen aus rund 4500 Jahren auf der Basis der Newcombschen Theorie der Sonnenbahn und der Mondtheorie von Brown. Einfache Orientierungskarten zum ungefähren Verlauf der Zentraallinie.
- [40] T. Oppolzer; *Canon der Finsternisse*; Denkschriften der Math.-Naturw. Classe der Kaiserlichen Akademie der Wissenschaften Bd. 52, Wien (1883); Dover Publications, Inc., New York (1962).
Elemente der Sonnen- und Mondfinsternisse im Zeitraum -1207 bis 2163. Karten zum Verlauf der Zentraallinie bei totalen Sonnenfinsternissen.

Das Buch *Astronomical Algorithms* von J. Meeus [8] stellt verschiedene Methoden vor, die sich insbesondere zur schnellen Vorhersage und Beurteilung möglicher Finsternistermine eignen. Fertige Tabellen aller Arten von Sonnenfinsternissen findet man in [9]. Die klassische Methode zur Berechnung von Sonnenfinsternissen mit Hilfe der Besselschen Elemente ist im *Explanatory Supplement* [4] und bei Mueller [10] beschrieben.

Aktuelle Vorhersagen von Sonnenfinsternissen bietet die Web-Seite
<http://planets.gsfc.nasa.gov/eclipse/eclipse.html>
von Fred Espanak.

Zu Kapitel 10 (Sternbedeckungen)

- [41] J. Robertson; *Catalog of 3539 Zodiacial Stars for the Equinox 1950.0*; Astronomical Papers of the American Ephemeris, vol. X, part 2; Washington (1917).

Referenzkatalog für die Vorhersage und Auswertung von Sternbedeckungen. Enthält ekliptiknahe Sterne bis etwa zur neunten Größenklasse, die vom Mond bedeckt werden können.

Die Vorhersage von Sternbedeckungen wird in verschiedenen Lehrbüchern der sphärischen Astronomie (zum Beispiel Green [5]) und im *Explanatory Supplement* [4] beschrieben. Nähere Einzelheiten zur Untersuchung der Mondbahn und der Erdrotation werden in dem Buch *Spherical and practical astronomy* [10] behandelt. Aktualisierte Sternpositionen können dem PPM Katalog [47] entnommen werden.

Zu Kapitel 11 (Bahnbestimmung)

- [42] C. F. Gauß; *Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections*; Dover Publications; New York;

Englische Übersetzung der „*Theoria motus corporum coelestium*“, in der Gauß seine Bahnbestimmungsmethode beschreibt. Das Buch ist im wesentlichen von historischem Interesse, eignet sich jedoch weniger als Lehrbuch der Bahnbestimmung.

- [43] H. Bucerius; *Bahnbestimmung als Randwertproblem I-V*; Astronomische Nachrichten, vol. 278, 280, 281, 282 (1950-1955);

Folge von fünf Aufsätzen, in denen unter anderem die Eignung der gekürzten und der ungekürzten Gaußschen Bahnbestimmungsmethode untersucht wird. Eine Vielzahl von Beispielen zeigt die möglichen Probleme, die im Rahmen der Bahnbestimmung auftreten können.

- [44] D. L. Boulet; *Methods of orbit determination for the micro computer*; Willmann-Bell; Richmond, Virginia (1991).

Darstellung von Grundlagen und Methoden der Ephemeridenrechnung und Bahnbestimmung. Behandelt werden unter anderem die spezielle Störungsrechnung sowie die Bahnbestimmung nach Laplace, Olbers und Gauss. Die abgedruckten Basic-Programme sind auch auf Diskette erhältlich.

Eine zeitgemäße Einführung in die Bahnbestimmung bietet darüber hinaus das entsprechende Kapitel in *Himmelsmechanik I* von Bucerius und Schneider [2].

Zu Kapitel 12 (Astrometrie)

- [45] *SAO Star Catalog*; Smithsonian Astrophysical Observatory; Cambridge Massachusetts (1966);

Vierbändiger Sternkatalog mit Positionen, Eigenbewegungen und Helligkeiten von 258 997 Sternen. Epoche und Äquinoktium 1950.0. Die Genauigkeit der Sternpositionen liegt bei rund 1".

- [46] W. Dieckvoss; *AGK3 Star Catalogue of Positions and Proper Motions North of -2°5 Declination*; Hamburg-Bergedorf (1975).

Umfangreicher Positions katalog des nördlichen Sternhimmels für astrometrische Aufgaben. Basierend auf photographischen Himmelsdurchmusterungen der Sternwarten Bonn und Hamburg-Bergedorf. Äquinoktium 1950.0.

- [47] S. Röser, U. Bastian; *Catalogue of Positions and Proper Motions*; Astron. Astrophys. Suppl. Ser. 74, 449 (1988).

S. Röser, U. Bastian; *PPM Star Catalogue*; Spektrum Akademischer Verlag, Heidelberg (1991).

U. Bastian, S. Röser; *Catalogue of Positions and Proper Motions - South*; Astronomisches Rechen-Institut, Heidelberg (1993).

U. Bastian, S. Röser; *The Bright Stars Supplement to the PPM and PPM South Catalogue*; Revised Edition, Astronomisches Rechen-Institut, Heidelberg (1993).

S. Röser, U. Bastian, A. V. Kuzmin; *The 90000 Stars Supplement to the PPM Star Catalogue*; Astron. Astrophys. Suppl. Ser. 105, 301 (1994).

Nachfolger des SAO Katalogs mit Positionen und Eigenbewegungen von insgesamt rund 470 000 Sternen der nördlichen und südlichen Halbkugel bezogen auf das Äquinoktium J2000 (FK5). Die verschiedenen Teile des Katalogs sind beim Astronomical Data Center der NASA unter

<http://adc.gsfc.nasa.gov/adc-cgi/cat.pl?/catalogs/1/1146/>

<http://adc.gsfc.nasa.gov/adc-cgi/cat.pl?/catalogs/1/1193/>

http://adc.gsfc.nasa.gov/adc-cgi/cat.pl?/catalogs/1/1206/

http://adc.gsfc.nasa.gov/adc-cgi/cat.pl?/catalogs/1/1208/

erhältlich. Die gedruckte Version des PPM Kataloges ist im Spektrum Akademie Verlag, Heidelberg, erschienen.

- [48] W. Tirion, B. Rappaport, G. Lovi; *Uranometria 2000.0*; Willmann-Bell; Richmond, Virginia (1987).

Atlas des gesamten nördlichen und südlichen Sternhimmels mit 473 Karten im Maßstab 1.85 cm=1° für das Äquinoktium 2000. Enthält Sterne bis 9m5 auf der Grundlage der Bonner Durchmusterung, der südlichen Bonner Durchmusterung und der Cordoba Durchmusterung.

Die Berechnung der Standardkoordinaten wird in vielen Lehrbüchern der sphärischen Astronomie behandelt (vgl. [5]). Hinweise zur Ausgleichsrechnung findet man in den Büchern zur numerischen Mathematik, wie [11] und [13].

Sachverzeichnis

- Aberration 150,207
 - planetare 124
 - stellare 125,249
- Abplattung 150
 - der Erde 185,187
 - der Planeten 139
- Accel 88
- Accumulate 266
- ADA 4
- Adams 93
- Additionstheoreme 118,160,169
- AddN 161
- AddSol 162
- AddThe 118,160,163
- Almagest 156
- Angle 9
- Anomalie
 - exzentrische 62,65,237
 - mittlere 65,97,111,112,115,118,237,238
 - wahre 60,69,73,111,232,237
- AOEcat 109
- Apparent 209
- Approximation 168
- Approximationsfehler 171
- Apsidenlinie 60
- Äquator 17,127
- Äquatorsystem 17
- Äquinoktium 7,18,73,124,219,236,249
 - B1950 20
 - des Datums 20,122,127,148,157,207
 - J2000 20
- Argument der Breite 73,158,236
- Argument des Perihels 72,237
- astorb.dat 108,278
- Astronomical Almanac 112
- Astronomische Einheit 63
- Atomuhr 41
- Auf- und Untergänge
 - geozentrische Höhen 46
 - Iterationsverfahren 47
- Ausgleichsproblem, lineares 265
- Ausgleichsrechnung 263
- Ausgleichung 268
- Azimut 35
- Bahn
 - nichtperiodische 60
 - parabelnahe 68
 - periodische 60
 - ungestörte 158
- Bahnbestimmung 231
 - Eindeutigkeit 243
 - nach Gauß 231
 - nach Laplace 231
 - verkürzte Gaußsche Methode 242
 - vollständige Gaußsche Methode 247
- Bahnelemente 59,231,238,242
 - der Planeten 89
 - Kleinplanet Ceres 106
 - oskulierende 97
- Bahnneigung 72,158,235
- Bahnparameter 60,232,236
- Barkersche Gleichung 64,69
- Bashforth 93
- Beleuchtungswinkel 147
- Bessel 203
- Besseljahr 20
- Bewegungsgleichung 86
- Bogemaß 65,164
- Brahe, Tycho de 156
- Brechungsgesetz 45
- Breite
 - ekliptikale 17,114
 - geographische 35,186,215,218
 - geozentrische 186,215
 - planetographische 139
 - planetozentrische 139
- Brennweite 259
- Bright 149
- Brown, E.W. 159
- Bucerius, H. 231
- C++ 94,233,279
 - GNU 281
 - Microsoft 280
- CaIDat 15
- Central 192
- Charliersche Grenzlinie 244
- Cheb3D 171,190
- Clenshaw, Algorithmus von 173

- Coco 7,28
- Comet 59,78
- Comet.dat 81,278
- Conjunct 213,219
- Contact 220
- Contacts 203
- Dämmerung 46
- Datum
 - Julianisches 14
 - Modifiziertes Julianisches 14
- Ddd 9
- DE 93
- Deklination 18,186
- Differentialgleichung 91
- Differenz ET-UT 42,188
- Beobachtung 228
- Polynomapproximation 188
- DMS 9
- Dreiecksfläche 232,236,241
- Durchmesser
 - scheinbarer 57,150
- Ebene, unveränderliche 135
- Ebenengleichung 240,241
- EccAnom 66
- Ecl2EquMatrix 19
 - Anwendungsbeispiel 19
- Eclipse 192,193
- EclTimer 203
- Eigenbewegung 206,219
- Ekliptik 17,73,111,127,156,179,205,235,249
 - mittlere 117
 - Pol der 17,156
 - Schiefe der 18,127
- Elements 99,238
- Ellip 66
- Ellipse 60,237
 - Fläche 61
 - Sektorfläche 62
- Elongation 146
- Encke 244
- Entfernung 239,241
 - geometrische 76,124
 - geozentrische 45
- Ephemeriden, physische 135
- Ephemeridenzeit 41,43,122,159,187,216
- Epoche 207,219
 - J1900 19
 - J2000 19
 - julianische 19
- Epoche J2000 122,159
- epsilon 279
- Equ2EclMatrix 19
- Equ2Hor 37
- Anwendungsbeispiel 41
- EquStd 262
- Erdachse 17
 - Lage der 126
 - mittlerer Pol 127
 - wahrer Pol 127
- Erdbahn 18,117
 - Mittelpunktsgleichung 113
 - Störungen 27
- Erdbahnlösung 243
- Erddrehung 40,125
- Erde
 - Abplattung der 215
 - Äquatorradius 186
 - Erdumlauf 125
 - ETminUT 188,216
 - Evektion 156
 - Examine 217,219
 - Exzentrizität 60,232,236
- F 95,233
- Filmebene 260
- FindEta 234
- Fit 172
- Flächensatz 59,61,232
- FORTRAN 4,93
- Foto 268
 - Foto.dat 267,278
 - Fotografie 259
- Fourierreihe 112
- Frac 8
- Frühlingspunkt 17,72,73,127,156,187,207
 - mittlerer 127
 - wahrer 127
- Fundamentalebene 184,201,214,216
- Funktionen
 - hyperbolische 64,233
 - Stumpffsche 70
 - trigonometrische 70,115,118,160
- Gauss 249
- Gauß C.F. 231,245
- Gauß-Lagrangesche Gleichung 245
- Gauss.dat 256,278
- GaussMethod 249
- Gaußsche Vektoren 72,75,111,235
- GaussVec 75
- Geschwindigkeit 126
- GetAccuracy 279
- Givens-Rotationen 265
- Gleichung
 - Barkersche 64,69
 - jährliche 156
 - Keplersche 62,111,113,237
- Gleichungssystem

- Dreiecksgestalt 264
- lineares 263
- überbestimmtes 263
- GMST** 40,187
- Goffin, E. 112
- Gordon 93
- Gradmaß 8,65,157
- Gravitationsgesetz 59,86,156
- Gravitationskonstante 86
- Gaußsche 63
- Greenwich, Meridian von 187
- Größenklassen 148
- Halbachse 60,111,232,237
- Halbschatten 181,205
- Durchmesser 182,192
- Hansensche Näherung 234
- Helligkeit, scheinbare 148
- Hill, Tafeln von 112
- Himmelsäquator 207
- Himmelspol 17,35,135
- Höhe 35
 - geozentrische 45
 - topozentrische 45
- Hor2Equ** 37
- Horizont 35,186
- Horizontalparallaxe 45,57
- Horizontsystem 35,37
- HypAnom** 67
- Hyperb** 68
- Hyperbel 60,237
- ILE_Pert** 160
- Illum** 147
- Improved Lunar Ephemeris 159
- Integ** 95
- Integration
 - numerische 91,111
- Interpolation
 - inverse 48
 - lineare 210
 - quadratische 48,217
- Interpolationspolynom 168
- Intersect** 192
- Intrp** 95
- Jahreslänge 15,42
- Jahrhundert 19
 - gregorianisches 19
 - julianisches 19,122,159
- JAVA** 4
- JupiterPos** 117,120,124
- Kalender, julianischer 15
- Kalenderreform, gregorianische 15
- Kegelschnittsgleichung 60,113,236,237
- Kepler** 75
- Kepler, Johannes 59
- Keplerbahn, ungestörte 111
- Keplerbewegung 112,117,240
- Keplerellipse, ungestörte 112
- Keplergleichung 62,69,111,113,237
 - für Hyperbelbahnen 64
 - Iteration 65,67
 - Startwert 66,67
- Keplersche Gesetze 59
 - drittes 63,238
 - erstes 59
 - zweites 59,232
- KepPosition** 91
- KepVelocity** 127
- Kernschatten 179,181,205
 - Durchmesser 182,189,192
 - Ort auf der Erde 184
- Knotenlänge 72,235
- Knotenlinie 158
- Komet Bradfield 259
- Kometenbahnen 59
 - Geschwindigkeit 69
- Konjunktion 206,210
 - iterative Bestimmung 210
- Konjunktionsraum 244
- Kontaktzeiten 200
- Koordinaten
 - äquatoriale 7,17,57,185,249,263
 - astrometrische 77,124,249
 - ekliptikale 7,17,72,75,249
 - Fundamentalebene 215
 - geographische 139
 - geometrische 124
 - geozentrische 7,24,44
 - heliozentrische 7,24,75,76,122,135,155
 - kartesische 10,36
 - mittlere 207
 - perifokale 73
 - planetographische 135,139
 - retardierte 77
 - scheinbare 124,127,207,208
 - sphärische 231,261
 - topozentrische 44
 - wahre 207
- Koordinatensysteme 7
- Kreisbahn 62
- Kreuzprodukt 97,235
- Kulmination 57
- Lagebahnelemente 72,235
- Lagrange 245
- Lagrange-Interpolation 169
- Länge
 - ekliptikale 17,113,179

- geographische 186,187,218
- heliographische 140,151
- planetographische 140
- planetozentrische 140
- Länge des Perihels 72,237
- Lichtgeschwindigkeit 76,125,207,248
- Lichtlaufzeit 76,124,150,192,248
- Linux 275,278
- Lokale Umstände 200
- Luna 174

- MarsPos** 124
- Maschinengenauigkeit 279
- Mat3D** 13
- Mayer, Tobias 156
- Meeus, J. 112
- Mehrschrittverfahren 92
- MercuryPos** 124
- Meridian 35,262
- Methode der kleinsten Quadrate 263
- MiniMoon** 38,158
- MiniSun** 39
- Mittelpunktsgleichung 112,156
- Mjd** 14
- Modulo** 8
- Monat
 - anomalistischer 156
 - synodischer 156
- Mond 155,205
 - Anziehungskraft der Erde 155
 - Anziehungskraft der Sonne 155
 - äquatoriale Koordinaten 168
 - aufsteigender Knoten 156
 - Bahnneigung 156,179,213
 - ekliptikale Breite 163
 - ekliptikale Länge 163
 - Entfernung 163,182
 - Ephemeride 174
 - große Halbachse 156
 - Höhe 43
 - Horizontalparallaxe 163
 - Keplerbahn 155
 - Knotenwanderung 127,156,179
 - Koordinaten 43,190
 - mittlere Anomalie 157
 - mittlere Bewegung 156
 - mittlere ekliptikale Länge 157
 - mittlere Elongation 157
 - mittlere Länge 157
 - mittlerer Knotenabstand 157
 - planetare Störungen 163
 - räumliche Bewegung 155
 - säkulare Störungen 156
 - Schatten 179,205
 - Störung durch die Sonne 155

- Störungsterme 159
- wahre ekliptikale Länge 157
- MoonEqu** 168,191
- MoonPos** 38,159,164

- Nautical Almanac Office 159
- NeptunePos** 124
- Neumond 179
- Neumondzeitpunkt 192
- Newcomb, Tafeln von 112
- Newton Isaac 59,156
- Newtonverfahren 65
- Nordol 135
- Nullmeridian, Lage des 141
- numeric_limits** 279
- Numerische Integration 91
- Numint** 100
- Numint.dat** 100,278
- Nutation 127,150,191,207
 - in Länge 127
- NutMatrix** 128

- Objektiv 259
- Observer** 203
- Occult** 205,220
- Occult.dat** 219,278
- Opposition 244
- Optische Achse 260
- Orient** 142
- Orkisz.dat** 257
- Ort
 - geometrischer 76
 - geozentrischer 239
 - heliozentrischer 239
- Ortssternzeit 40,47

- Parab** 71
- Parabel 60,64,237
- Parallaktischer Winkel 202
- Parallaxe 45
- Pegasus** 211
- Pegasus-Verfahren 180,204,211
- Penumbra 182
- Perigäum 156
- Perihel 60
- Periheldistanz 61,64,237
- Periheldurchgang 62
- Perihelzeit 237,238
- Pert** 119
- PertPosition** 57,91,123,150
- Phase 146
- Phasenwinkel 146
- Phases** 181
- PhasesFunc** 180
- Phys** 150,151

- Planeten
 - Äquatorradien 139
 - Koordinaten 89
 - Massen 87
- Planetenbahn 111
 - Reihendarstellung 114
- Planpos** 129
- Planrise** 57
- Plattenkonstanten 263,268
- Plejaden 206,219
- PlutoPos** 124
- Polarkoordinaten 10,17
- Polynomentwicklung 170
- PosAng** 138,147
- PosAngles** 204
- Position, scheinbare 126
- Positionswinkel 200,217,220
 - der Rotationsachse 136
 - der Sonne 147
- PPM-Katalog 206,228,259
- PPM.bin** 273,278
- PPM.dat** 273,278
- PPMbin** 273
- PPMcat** 273,274
- Präzession 7,20,73,127,150,157,207
 - in Länge 22
 - Rechenbeispiel 24
- Präzessionswinkel 22
- PrecMatrix.Ecl** 23
- PrecMatrix.Equ** 23
- Ptolemäus 156
- Quad** 49,203,217
- Radius, scheinbarer 46
- RA_Diff** 212
- ReadCatalogue** 209
- Reduktion auf die Ekliptik 113
- Reformdatum 15
- Refraktion 45
- regula falsi 180,204,210
- Reihenentwicklungen 111
- Rektaszension 18,186,187,206
- Relativitätstheorie 41,126
- Residuen 264
- Rotation** 145
 - direkte 136
 - retrograde 136
- Rotationsachse, Orientierung der 136
- Rotationsellipsoid 185,186,215
- Rotationssysteme 140
- SAO-Katalog 20,206
- Saturn, Ringsystem 148
- SaturnPos** 124
- Scaliger
 - Joseph Justus 14
 - Julius 14
- Schaltsekunde 42
- Schalttag 15
- Schrittweite 93
- Sekantenverfahren 234
- Sektor-zu-Dreieck-Verhältnis 231
 - Iteration 234
- Sektorfläche 232,241
- Sekunde 41
- ShadowDist** 203
- Shampine 93
- Shape** 139
- Site** 216
- Skalarprodukt 240
- Solve** 267
- SolveGL** 246,249
- SolverDE** 94
- SolverLSQ** 265
- Sonne
 - beobachteter Ort 192
 - geometrischer Ort 192
 - mittlere Anomalie 157
- Sonnenfinsternis 179,205
 - Dauer 189,192
 - Größe 202
- Kontaktzeiten 200
 - lokale Umstände 200
 - nichtzentrale Phase 185
 - partielle 185
 - Phasen 185
 - Positionswinkel 200
 - ringförmige 182,204
 - totale 179
 - zentrale Phase 184
- Zentrallinie 192
 - Sonnenkoordinaten 25,76,190,240
- Sonnentag 39
 - Sonnenzeit 39,41,47
- Standardkoordinaten 262,268
- Star** 208
- Start** 249
- State** 89,127
 - Stationskoeffizienten 205,217,220
- StdEqu** 262
- Step** 95
 - Sternbedeckung 44,168,205
 - Kontaktzeiten 216
- Sterntag 40,47
 - Sternzeit 39,42,57,135,187,215
- Steyart, C. 112
- Störbeschleunigung 87
- Störungsrechnung 85

- Störungsterme 112,115
–periodische 111,115
–säkulare 117,124
Stumpff 71
Stumpff, Methode von 68
Stundenmaß 36
Stundenwinkel 36,39,40,47
SunEqu 191
SunPos 25,38,124
Sunset 44,51
- Tagbogen 47
Tageslänge, mittlere 42
Term 120,161
Terminator 147
Totalitätsdauer 189
Tschebyscheff-Approximation 168,171
Tschebyscheff-Entwicklung 190,212
–Auswertung 173
Tschebyscheff-Polynome 169
–algebraische Darstellung 170
–Nullstellen 170
–Rekursionsbeziehung 169
–trigonometrische Darstellung 169
- Umlaufzeit 63,238
Ungleichheit, große 156
UranusPos 124
- Value** 173
- Variation 156
Vec3D 11
Vektoren
–Gaußsche 72,75,111,235
VenusPos 124
Verfinsterungsgrad 182
Vergleichssterne 259
- Weltzeit 41,43,122,187,216
–koordinierte 42
Windows 275,277
- ZC.dat** 228
Zeit 41
–dynamische 41,42
–dynamische baryzentrische 41
–dynamische terrestrische 41
Zeitdifferenz ET-UT 42
Zeitmaß 47
Zeitzählungen 41
Zeitzonen 42,216
Zenit 35,45
Zentralkraft 60
Zentrallinie 190
Zentralmeridian 140
Zirkumpolarstern 47
Zodiakalkatalog 206
Zustandsvektor 91
Zweikörperproblem 111,232
Zwischenzeit 232,236,241

**Rücknahme oder Umtausch
nur mit ungeöffneter Datenträgerverpackung**

Systemvoraussetzungen:

- Windows 95/98/NT oder SuSE Linux 6.0
- 133 MHz Pentium Prozessor oder vergleichbar
- Arbeitsspeicher \geq 16 MB
- Freie Festplattenkapazität \geq 10 MB (150 MB empfohlen)
- Optional: Microsoft Visual C++ 5.0 oder GNU C++ 2.91