Lichtkurven-Analyse mit Python

Mike Kretlow

Einleitung

Die Erfassung und Auswertung von Asteroiden-Lichtkurven ist ein spannendes und wissenschaftlich wertvolles Betätigungsfeld für Amateure. Betrachtet man den Umfang eines Jahrgangs des Minor Planet Bulletin, dem Hauptpublikationsorgan für Asteroiden-Lichtkurven, dann kann man feststellen, dass dieser sich in den letzten zehn Jahren in etwa verdreifacht hat. Ein Grund dafür mag sein, dass die reine Astrometrie und die Suche nach Kleinplaneten für viele Amateure, insbesondere mit kleineren und mittelgroßen Instrumenten, in der letzten Dekade unattraktiver geworden ist.

In diesem Beitrag wird die Reduktion von Zeitreihen-Photometrie von Kleinplaneten mit Hilfe freier Open-Source Software als Alternative zu einigen kommerziell erhältlichen (Closed-Source) Programmen vorgestellt. Die Motivation dazu ist weniger ökonomischer Natur, sondern vielmehr dass tiefere Eindringen in die Thematik bei selbst entwickelter Software und bei Open-Source Produkten, der Spaß am Programmieren und nicht zuletzt die Freiheit, selbst geschriebene Programme (oder Skript-Sammlungen) ganz nach den eigenen Wünschen und Bedürfnissen anpassen zu können (bis hin zur vollautomatischen Auswertung).

Software zur Auswertung

Viele Wege führen nach Rom. Das trifft auch auf die Reduktion von CCD-Beobachtungen (Photometrie) und die nachfolgenden Lichtkurven-Analyse zu. Auf die Photometrie an sich wird im nachfolgenden nicht im Detail eingegangen. Auf dem Markt existieren mehrere Programme, die eine photometrische Auswertung von CCD-Aufnahmen ermöglichen. Wenige sind auch in der Lage dies automatisiert (im Stapelbetrieb) für bewegliche Objekte zu tun. Manche erlauben Ringblenden-Photometrie relativ zu einem (oder ggf. mehreren) Vergleichssternen, andere liefern Messwerte für alle identifizierten Objekte auf einer Aufnahme. Für die Lichtkurven-Analyse ist es nicht zwingend, absolute Helligkeits-Messungen zu verwenden. Wesentliche Ergebnisse wie die ermittelte Rotationsperiode des Kleinplaneten und die maximale Amplitude der gemessenen Lichtkurve sind davon unabhängig. Auch mit dem in der Astrometrie weit verbreiteten Programm Astrometrica kann die Photometrie erfolgen. Die erzeugten Log-Dateien können für die nachfolgende Auswertung genutzt werden. Leider bietet

Astrometrica keinen Kommandozeilen-Modus oder Stapelbetrieb, so dass die Auswertung von hunderten von Bildern (in kleinen Paketen von jeweils nur einigen Bildern) recht zeitraubend ist. Alternativ dazu kann die frei erhältliche Software IRIS von Christian Buil verwendet werden [1]. IRIS ist auch in der Lage, einen Asteroiden automatisch auf einer Sequenz von Bildern zu photometrieren. Beide Programme sind nur für MS-Windows verfügbar, laufen aber auch problemlos auf Linux mit Hilfe der Windows-kompatiblen Laufzeitumgebung Wine [2]. Selbstverständlich ist die Aufzählung damit nicht vollständig, diese beiden Programme werden hauptsächlich vom Autor verwendet.

Was die Lichtkurven-Auswertung (mathematisch gesprochen: Zeitreihen-Analyse) betrifft, so findet man im Bereich der Veränderlichen-Beobachtung diverse Programme, die dazu genutzt werden können. Zum Beispiel das kommerzielle Programm Peranso [3]. Das MS-Windows Programm implementiert zahlreiche verschiedene Algorithmen zur Perioden-Analyse (die z.T. auch für Asteroiden-Lichtkurven genutzt werden können) und wurde in den letzten Versionen für die Auswertung von Exoplaneten Transits erweitert. Daneben wurde auch der sog. FALC-Algorithmus implementiert, der häufig als Referenz für die Auswertung von Asteroiden-Lichtkurven genannt wird [4]. Abbildung 1 zeigt das Ergebnis einer solchen Auswertung mit Peranso.

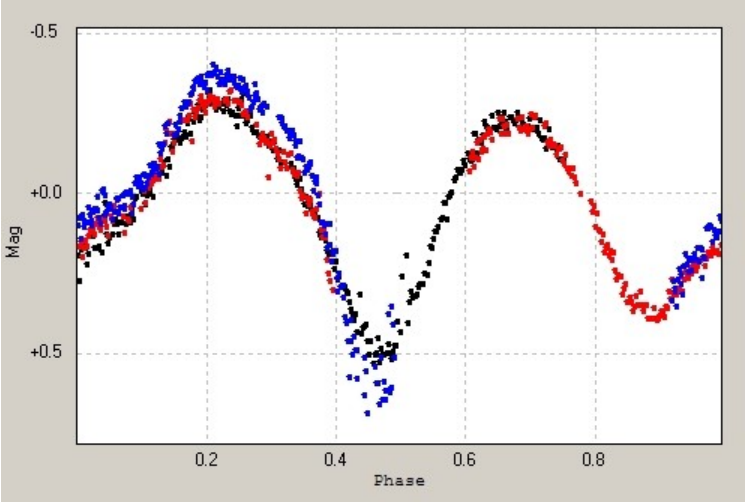

Abbildung 1: Phasenlichtkurve des Asteroiden (24445) 2000 PM8, aus drei Beobachtungsnächten mit Peranso generiert: 2013 Sept. 19 (schwarz), 20 (rot), 25 (blau).

Es soll nicht unerwähnt bleiben, dass mit MPO Canopus / PhotoRed ein kommerzielles MS-Windows Programm verfügbar ist, das sich ganz auf die Astrometrie, Photometrie

und insbesondere Lichtkurven-Analyse von Asteroiden konzentriert. Der überwiegende Teil der im Minor Planet Bulletin publizierten Beiträge zitiert Canopus als verwendete Auswertesoftware.

Python (und Julia)

Die freie Programmiersprache Python [5] hat in den letzten Jahren in der Wissenschaft und somit auch in der (Amateur- und Profi-) Astronomie beträchtlich an Beliebtheit gewonnen. So existieren große Paketsammlungen für mathematisch-wissenschaftliche Anwendungen und für die professionelle Visualisierung [6]. Neben Fortran und spezieller (kommerzieller) Software zur Datenanalyse und zur Lösung mathematischer Probleme wie MATLAB und IDL, hat sich Python etablieren können – obwohl Python als einfach zu erlernende, allgemeine Skriptsprache und nicht mit Blick auf mathematische Anwendungen entwickelt wurde. Die "Schwächen" in diesem Bereich (insbesondere auch in puncto Geschwindigkeit) wurden durch später entstandene numerische Pakete ausgeglichen (z.B. NumPy). Die noch sehr junge Programmiersprache Julia [7] wurde und wird hingegen mit genau diesem Anspruch entwickelt, nämlich als (High-Performance) Programmiersprache für das numerische und wissenschaftliche Rechnen. Sie hat das Potential einen vorderen Platz in mathematisch-wissenschaftlichen Applikationen einzunehmen, aber die Zeit wird zeigen, ob dies gelingen wird. Schließlich hängt die Akzeptanz auch von der Verfügbarkeit von entsprechenden Paketsammlungen (Bibliotheken) ab, um bestimmte mathematische Probleme innerhalb seiner Anwendung lösen zu können, ohne das Rad neu erfinden zu müssen. Andererseits ist es in Julia bereits nativ sehr einfach möglich, in C oder Fortran geschriebene Bibliotheken zu nutzen und auch die Anbindung an Python ist über das Paket PyCall gegeben. Insofern ist das, was im nachfolgenden als Python-Implementierung vorgestellt wird, ebenfalls in Julia möglich, auch wenn sich dieser Beitrag auf Python beschränkt.

Sowohl Python als auch Julia sind für alle drei große Plattformen (MS-Windows, Linux / Unix, Mac OS X) frei als Open-Source verfügbar.

Auswerte-Prozess

Bevor wir uns mit einer konkreten Implementierung beschäftigen, sollen zunächst einmal die wesentlichen Arbeitsschritte einer solchen Auswertung skizziert werden:

1. Die Photometrie der CCD-Beobachtungen kann, wie eingangs erwähnt, sowohl mit IRIS, als auch mit Astrometrica oder einem anderen Programm geschehen,

welches vom Beobachter bereits verwendet wird. Letztendlich muss es in der Lage sein, die Messungen in Form einer Textdatei zu erzeugen, etwa in der Form: JD, Magnitude. Ggf. kann dies noch um eine weitere Spalte für die Messunsicherheit (ErrMag) erweitert sein (wie zum Beispiel bei Astrometrica der Fall). Ob und wie diese Angabe verwertet wird, soll hier nicht weiter diskutiert werden.

2. Messwerte einlesen (JD, mag, …).
3. Helligkeiten auf Einheitsdistanz (r = Δ = 1 AE) reduzieren.
4. Aus Absolutmessungen Relativwerte generieren, bezogen auf den Mittelwert der Nacht.
5. Korrektur der Lichtlaufzeit.
6. Periodensuche.
7. Visualisierung (Messungen, Phasen-Lichtkurve).

Schritte 2 bis 7 können in einem einzigen oder in mehreren Skripten implementiert sein. Für die Schritte 3 und 5 wird eine Ephemeride des Asteroiden für den Beobachtungszeitraum benötigt (jedoch nur die Entfernung zur Erde und zur Sonne für jeden Beobachtungspunkt). Dies kann z.B. dadurch erfolgen, dass man das Paket PyEphem [8] verwendet und die Ephemeride in seinem Skript für jeden Beobachtungspunkt direkt berechnet oder aber man generiert sich eine Ephemeride mit anderen Mitteln (z.B. dem Ephemeridenservice des Minor Planet Centers), speichert sie in einer Textdatei ab und nutzt diese in seinem Skript (einlesen und Werte interpolieren).

AsPyLib

Das von Jerome Caron entwickelte Paket AsPyLib [9] ist eine Sammlung von Python-Skripten zur Verarbeitung von FITS-Bildern und zur Photometrie von veränderlichen Sternen und Asteroiden. Grundsätzlich ist das Paket in seinem Gesamtumfang in der Lage, eine Sequenz von CCD-Aufnahmen im FITS-Format halbautomatisch zu photometrieren (auch bewegliche Objekte wie Asteroiden) und auszuwerten, bis hin zur Periodensuche und der Visualisierung der Ergebnisse bzw. Messungen (Plots mit Hilfe der matplotlib Paketsammlung). Dazu bedarf es aber einiger Einarbeitung bzw. Kenntnisse in der Feinjustierung diverser Parameter. Es ist keine fertige "Plug-and-Play" Lösung. Hat man die Photometrie aber mit anderen Programmen bereits erledigt, kann man sich einfach der Teilmodule für die reine Periodensuche in eigenen Skripten (bzw. angepassten Versionen der im Paket enthaltenen Beispiele) bedienen.

Im Unterordner 06_cdr findet man Beispielskripte, die zusammen die gesamte Lichtkurven-Analyse einschließlich der Visualisierung bewerkstelligen. Nachdem man sich ein wenig mit diesen Skripten befasst und sie den eigenen Gegebenheiten angepasst

und ggf. um weitere Funktionalitäten erweitert hat (z.B. der publikationsreifen Generierung von Grafiken mit Hilfe des Paketes matplotlib), erhält man Resultate ähnlich den Abbildungen 2–5, welche vom Autor zur Publikation im Minor Planet Bulletin verwendet wurden.

Zunächst wird nach der Periode (Rotationsdauer) des Asteroiden gesucht. Dazu verwenden wir das Skript Lightcurve_searchperiod.py aus dem Unterordner 06_cdr. Neben den Datenpfaden zu den Messungen benötigt das Skript im wesentlichen Angaben zu dem Periodenbereich der „gescannt“ werden soll, also z.B. 2–20h in 0.05h-Schritten und die Ordnung *n* der Fourier-Polynome, die gefittet werden sollen (etwa $n = 4$ oder 6). Das Skript passt der Reihe nach mit Hilfe der Methode der Kleinsten Quadrate für jede vorgegebene Periode eine Fourier-Reihe an die Messwerte an und berechnet den verbleibenden Anpassungsfehler (fit error). Am Ende erhält man eine grafische Darstellung aller Anpassungsfehler (Abbildung 2), aus der man die Periode mit dem kleinsten Wert als wahrscheinlichste Periode ermitteln kann. Insbesondere dient diese Darstellung auch dazu, weitere, ebenfalls mögliche Perioden (mit ähnlich kleinen Anpassungsfehlern) zu identifizieren. Das angezeigte Fenster erlaubt u.a. darin zu zoomen und mit dem Cursor Messwerte zu ermitteln. Abschließend kann die Grafik mit dem Speichern-Button in verschiedenen Formaten abgespeichert werden (so wie in den nachfolgenden Abbildungen der Fall).

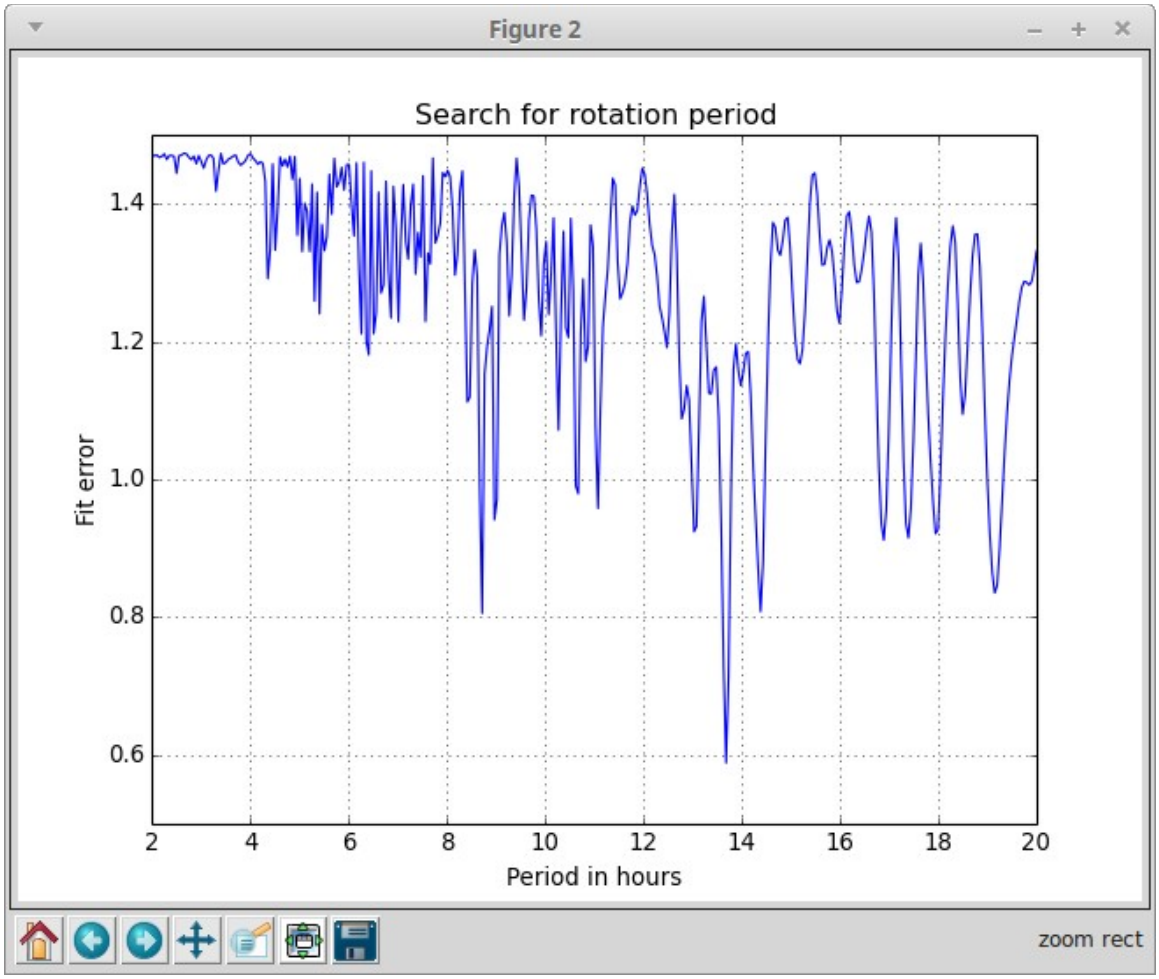

Abbildung 2: Periodensuche für den Asteroiden (30) Urania aus Messungen im Januar 2012. Gescannt wurde der Bereich P=2h bis P=20h in 0.05h-Schritten. Als wahrscheinlichste Periode ergibt sich hier jene mit dem kleinsten Anpassungsfehler (größter peak nach unten), also knapp 14h.

Mit einem zweiten Skript (Lightcurve_fitperiod.py) erfolgt die Suche nach der Periode analog zum ersten Skript, wobei jene Periode mit dem kleinsten Anpassungsfehler automatisch als Resultat nebst mittlerem Fehler und weiteren Ergebnissen (Amplitude etc.) angezeigt wird:

```
------ Fit results ------
Num. harmonics n =  6
Period (days)    =  0.570489058561 +/- 3.91604178262e-05
```

```
Period (hours)   =  13.6917374055 +/- 0.000939850027829
Amplitude (mag)  =  0.194335686818
Date of max mag  =  [ 2455933.75855565]
Date of min mag  =  [ 2455933.44509762]

--- Amplitudes (from observations) ---
Peak-to-Peak amplitude = 0.264
Min/Max observation    = -0.121 / 0.143
Mean (05) amplitude    = 0.194 +- 0.114
```

Dazu erhält man neben der zeitlichen Darstellung der Helligkeits-Messungen (verschiedene Nächte in verschiedenen Farben, Abbildungen 3 und 4) auch die wichtigste Grafik, eine Phasendarstellung der Messungen (Abbildung 5), die üblicherweise in der Publikation der Ergebnisse Verwendung findet.

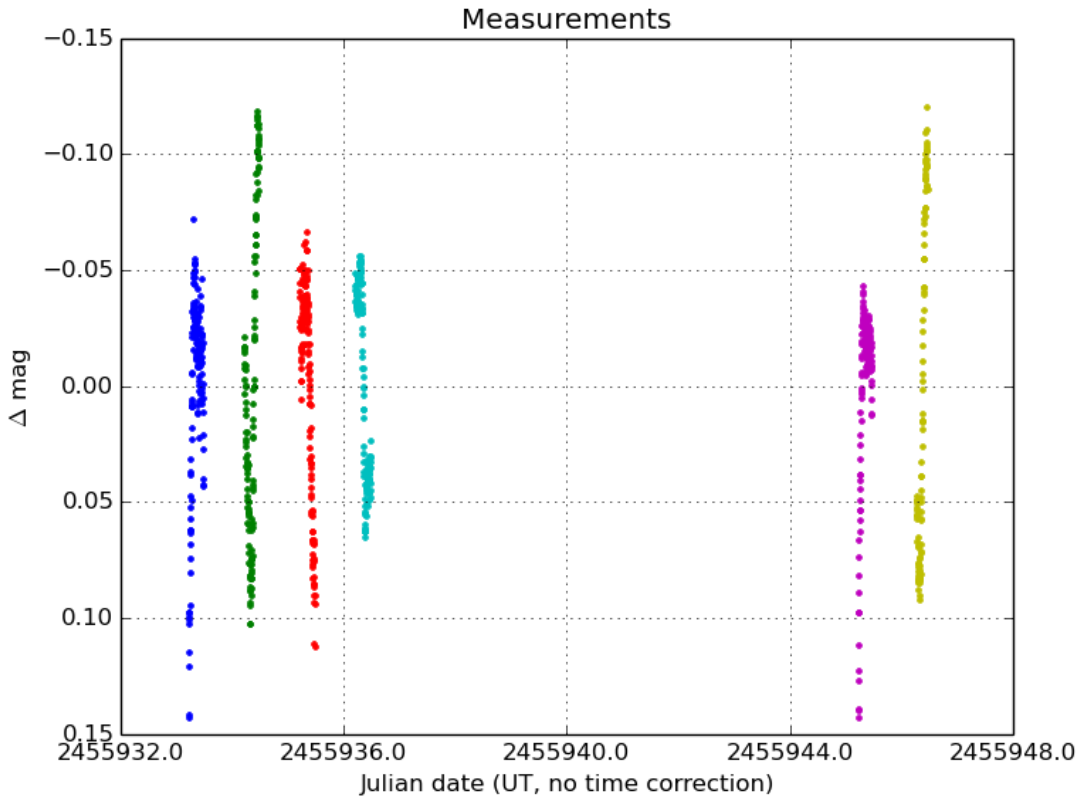

Abbildung 3: Zeitliche Darstellung der Helligkeitsmessungen am Asteroiden (30) Urania. Alle Messreihen wurden zunächst auf Einheitsdistanz reduziert und anschließend auf einen Mittelwert bezogen. Eine Korrektur der Lichtlaufzeit erfolgt in dieser Darstellung noch nicht.

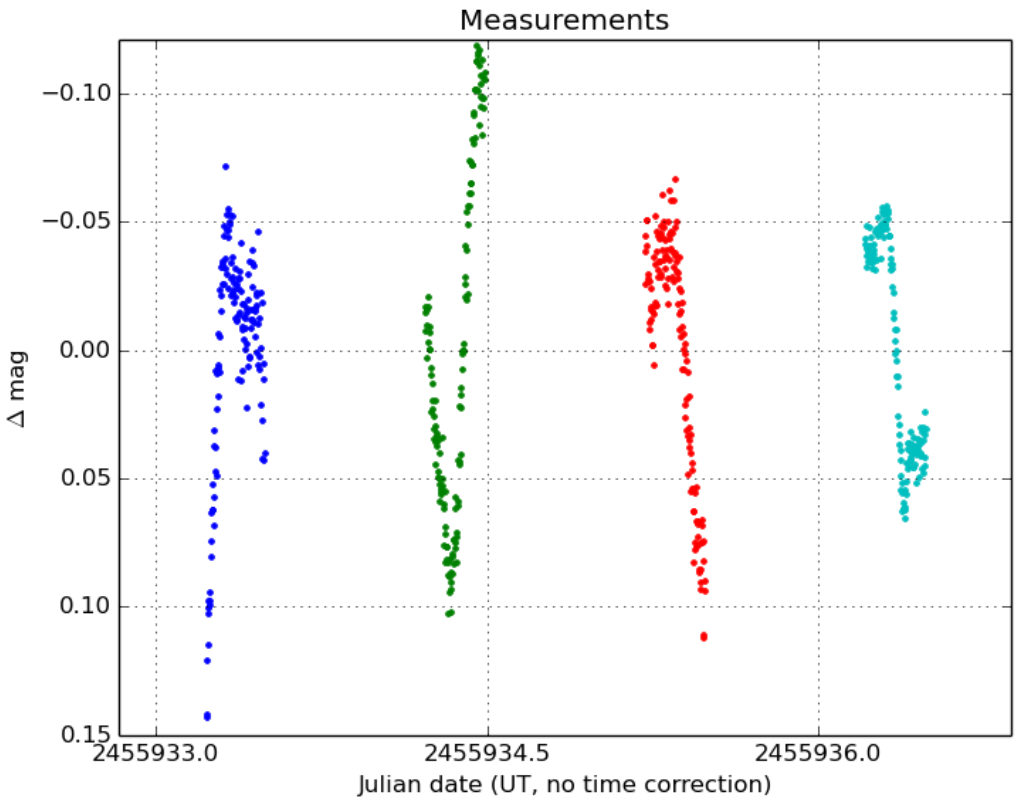

Abbildung 4: Vergrößerter Ausschnitt (Zoom) auf die ersten 4 Nächte aus Abbildung 3.

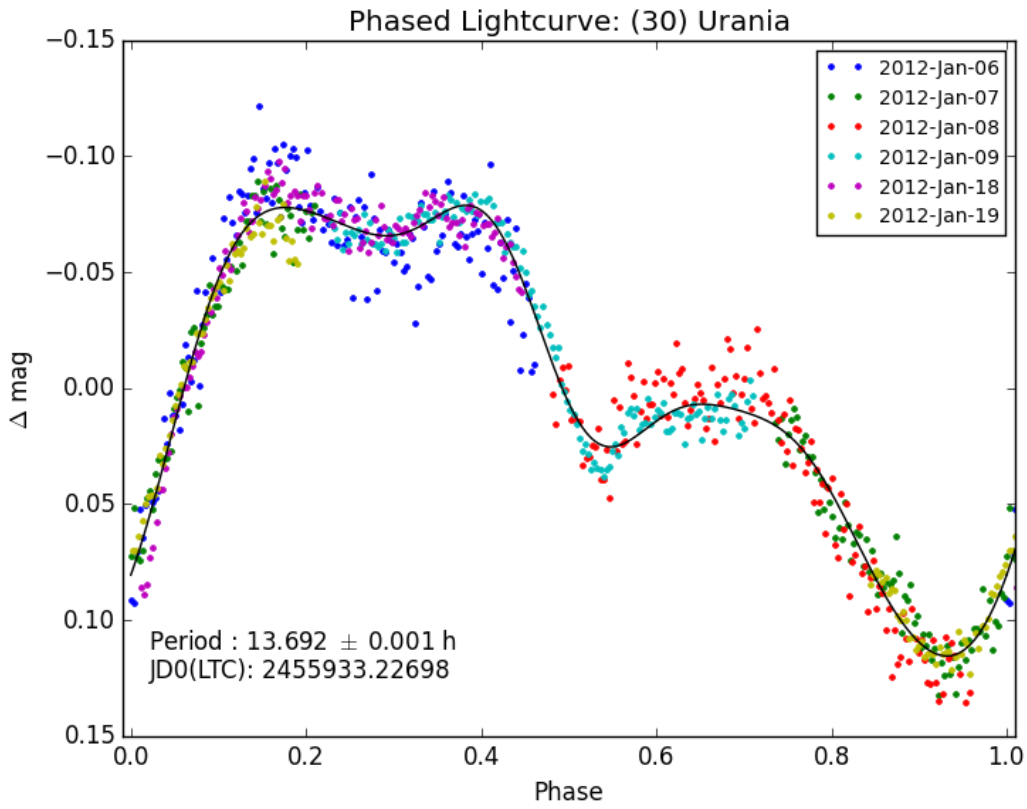

Abbildung 5: Phasendarstellung der Helligkeitsmessungen zur Epoche JD = 2455933.22698.

Schlussbetrachtung

Python bietet zusammen mit solchen Paketen wie NumPy, SciPy, matplotlib und ggf. Pandas fantastische Möglichkeiten, sich mit bescheidenem Programmieraufwand professionelle Werkzeuge zur Auswertung und Verarbeitung photometrischer Zeitreihendaten zu schaffen. AsPyLib ist hierfür ein gutes Beispiel. Man kann das Paket und die darin enthaltenen Anwendungs-Skripte sogleich einsetzen oder als Grundlage für Weiterentwicklungen nutzen.

Quellen- und Literaturhinweise

1: http://www.astrosurf.com/buil/us/iris/iris.htm
2: http://www.winehq.org/
3: http://www.peranso.com/
4: Harris, A.W. et al. (1989). Photoelectric observations of asteroids 3, 24, 60, 261, and 863. Icarus 77, 171-186.

5: http://www.python.org/
6: http://astropy.org/ | http://scipy.org/ | http://matplotlib.org/
7: http://julialang.org/
8: http://rhodesmill.org/pyephem/
9: http://www.aspylib.com/