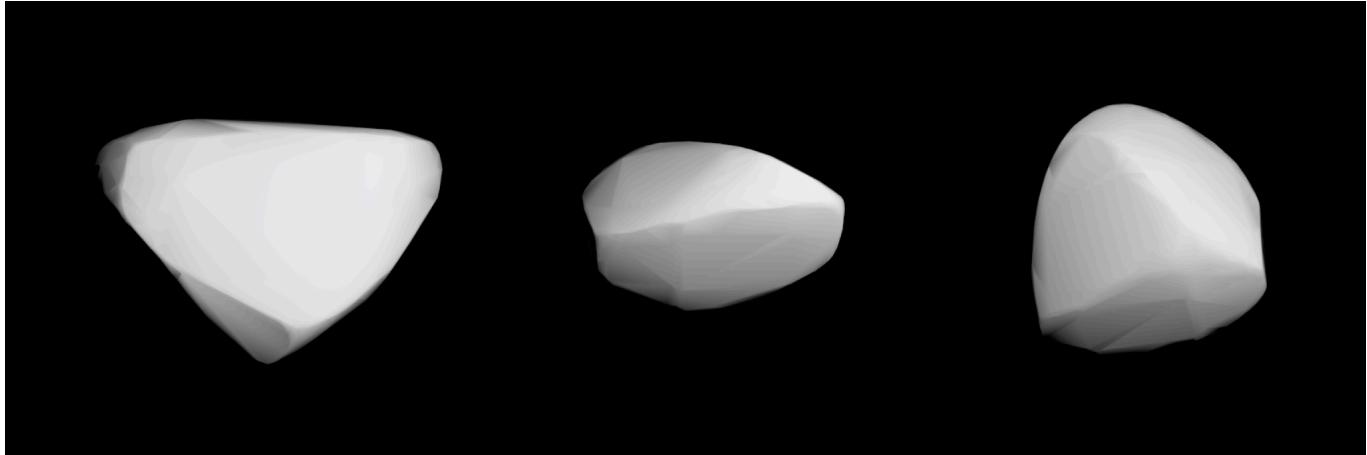


# Rotation Period, Pole Solution, Shape Model of Minor Bodies with DAMIT Code



University of Tokyo Jin BENIYAMA  
[beniyama@ioa.s.u-tokyo.ac.jp](mailto:beniyama@ioa.s.u-tokyo.ac.jp)

Last updated: November 13th, 2022

## 概要

近年光度曲線から小惑星形状を推定するためのソフトウェア DAMIT が広く用いられている。DAMIT のウェブサイトではこれまでの光度曲線観測結果を用いた数千天体の形状推定結果が掲載されている。DAMIT はコードがテストデータと共に公開されており誰でも無償で小惑星形状を再現できるにもかかわらずまとまったドキュメントはソフトウェアに付属する短い pdf のみである。DAMIT ではいくつかの手順で入出力ファイル形式を整える必要があるがその情報は見当たらない。形状推定が主な結果となる論文は少ないこともあり詳細に手法を述べる論文も少ない。また日本語の資料はほとんど見つからない。

このような背景から、誰でも形状推定(最終的にはよく見る 3 方向から多面体に近似された小惑星の図を作ること)ができるようになる資料が有用であると考え執筆を開始した。本稿の内容に問題がないことを確認したのちに英語へ翻訳予定である。

## To Do

- convexinv が何をしているのかを記述
- 参考文献を明記

# 目次

<b>概要</b>	1
<b>第 1 章 Lightcurve Inversion</b>	3
1.1 はじめに . . . . .	3
1.2 状況設定 . . . . .	3
1.3 Shape . . . . .	4
1.4 Scattering law . . . . .	5
1.5 計算 . . . . .	6
1.6 まとめ . . . . .	6
<b>第 2 章 準備</b>	7
2.1 ソフトウェアのインストール . . . . .	7
2.2 コンパイル . . . . .	8
<b>第 3 章 自転周期の決定</b>	9
3.1 準備 . . . . .	9
3.2 実行と描写 . . . . .	12
3.3 実行と描写 (MC 法) . . . . .	13
<b>第 4 章 自転軸の推定、形状モデルの作成</b>	15
4.1 軸推定 . . . . .	15
4.2 最終形状モデル推定の準備 . . . . .	19
4.3 形状モデルとモデルカーブの可視化 . . . . .	19

# 第1章

## Lightcurve Inversion

この章では `convexinv` で行われる lightcurve inversion (以下 LI)<sup>1</sup>について説明する。とにかく形状推定をしたい場合は次の章に移動すれば良いが、この章の内容を理解することを強く推奨する。本章の内容は主に Kaasalainen2001a, 2001b に沿う。これらの論文は小惑星のライトカーブから形状推定する方法を詳しく記述している。理論的な研究はすでに Kaasalainen1991a,b で行われているが、Kaasalainen2001a のイントロダクションの後半にも書かれている通り、本章の内容は Kaasalainen2001a,b で閉じている。

### 1.1 はじめに

LI は小惑星の形状の作成、その形状から期待される明るさの見積もり (モデル)、観測とモデルの残差の最小化を行う。初期の形状は 3 軸橢円体 (ellipsoid) で与えられ、それ以降は球面調和関数 (spherical harmonics) で表される。各面の形状は三角形 (triangle) で与える。形状の最適化は conjugate method (Press1994) で行われる。形状から頂点の位置を決める際は Minkowski minimization (Kaasalainen1992a, Lamberg1993) で行われる。Levenberg-Marquardt optimization scheme (Press1994) によって最終的な形状が決定する。本章ではこれらの流れを丁寧に説明する。

### 1.2 状況設定

ライトカーブインバージョンでは小惑星の形状を三角形 (triangle) の面をもつ多面体で表す (triangulation)。triangulation は、例えば octant procedure などで行う。低アルベド小惑星では多重散乱の寄与は小さいので、観測者からみてどの面が照らされているのか、が重要となる。

さて、長さ  $ds$  のパッチが太陽に照らされて見える時、明るさ全体に対する寄与は以下で表される。

$$dL = S(\mu, \mu_0) \varpi ds \quad (1.1)$$

$S$  と  $\varpi$  は scattering law、 $\mu$  はアルベドで、 $\mu = \vec{E} \cdot \vec{n}$ 、 $\mu_0 = \vec{E}_0 \cdot \vec{n}$  という関係を満たす。ただし  $\vec{E}$  と  $\vec{E}_0$  はそれぞれ観測者、太陽方向の単位ベクトルで、 $\vec{n}$  は面に垂直な単位ベクトル。Lambert law では  $S_L = \mu \mu_0$ 、Lommel-Seeliger law では  $S_{LS} = S_L / (\mu + \mu_0)$  となる。

ray-tracing は非常に単純に行われる。まず、各面 (facet) に対してその地平線よりも上にある頂点 (vertices) をチェックし、これらの頂点に接続したどの面が注目している面に向いているのかを確認する。これらは光を遮るものとなりうるので、先に見つけておくと、計算が速くなる。

実際の計算は各面ごとに行われる。 $\mu$  と  $\mu_0$  が正であれば「見える面」と判断される。注目している面の重心が他の面 (blocker) に阻まれている場合、見えない面として除外する (shadowing)。ただしこの面が見えている部分の中で非常に大きな割合を占める時、重心のみで判断して除外するのは不適となる。その場合は面内でいくつか点をのってそれぞれに対して見える、見えないを判断する。

以上の状況設定を図 1.1 に示す。

図 1.1

---

<sup>1</sup> convex inversion(CI) を書くこともある。

## 1.3 Shape

### 1.3.1 Spherical harmonics

LI では小惑星の形状を Spherical harmonics(球面調和関数) で記述する。

$$a_{lm} Y_l^m(\theta, \phi), \quad (1.2)$$

ただし  $a_{lm}$  は各球面調和関数の寄与を表す係数、 $Y_l^m(\theta, \phi)$  は degree of the harmonic  $l$ 、order of the harmonic  $m$ (ただし  $m \leq l$ ) の球面調和関数。(なぜ球面調和関数で表すのか。数学的に取り扱いやすい?)

DAMIT のデフォルトでは  $l = m = 6$  の Spherical harmonics が用いられる。LI のフリーパラメタの数  $M$  は形状のパラメタ ( $a_{lm}$ ) が支配的である。以下、 $l = m$  の場合を仮定する。絶対的な明るさを取り扱う場合、パラメタ数は  $(l + 1)^2$  である。一方、相対的な明るさを取り扱う場合(つまり入力が相対光度曲線の場合)、小惑星の絶対的なサイズは求められない。よってスケールを表す係数  $a_{00}$  が任意の値となりフリーパラメタでなくあるので、パラメタ数は  $(l + 1)^2 - 1$  となる。

図 1.2 に  $l = m = 6$  までの球面調和関数を示す。小惑星の形状はこれらをスケーリングした重ね合わせで表される。ちなみに Vokrouhlicky+2017 では観測数  $N$ 、自由度  $\nu$  から逆算したパラメタ数  $M$  は 87 となる。形状に関するパラメタ  $a_{lm}$  が LI のパラメタ数の大部分を占めることを考えると、おそらく  $l = m = 8$ あたりを使っていると、筆者は予想する。このような高次まで取り入れるべきかどうかはよくわからない。

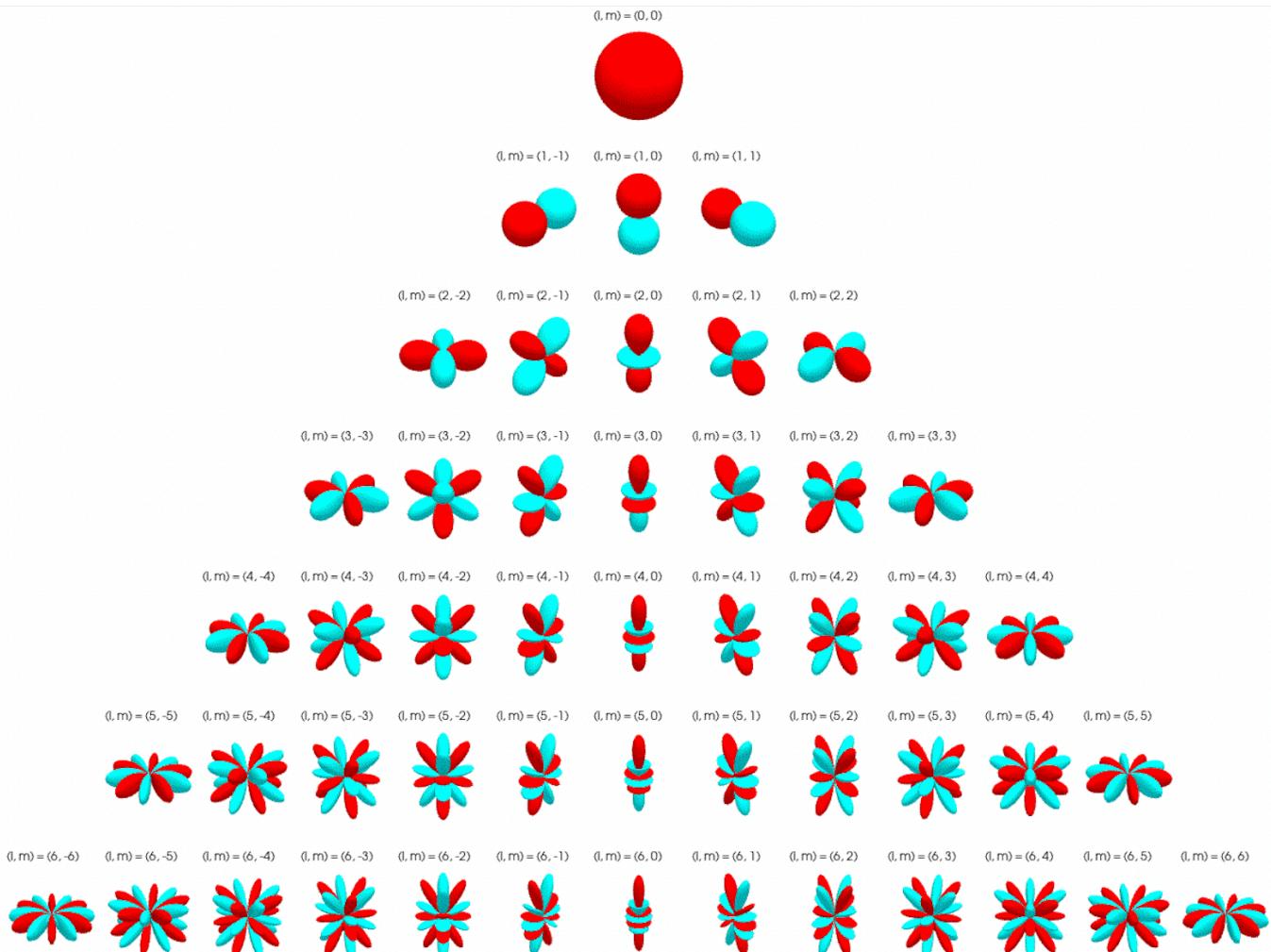


図 1.2 Twistar48 氏によるアニメーション [https://en.wikipedia.org/wiki/Table\\_of\\_spherical\\_harmonics#/media/File:Real\\_Spherical\\_Harmonics\\_Figure\\_Table\\_Complex\\_Radial\\_Magnitude.gif](https://en.wikipedia.org/wiki/Table_of_spherical_harmonics#/media/File:Real_Spherical_Harmonics_Figure_Table_Complex_Radial_Magnitude.gif) (CC BY-SA 4.0) を一部切り取った。

## 1.4 Scattering law

### 1.4.1 Single and multiple scattering

LIにおいて、single scattering、multiple scattering の寄与の大きさは weight factor  $c$  で表される。single scattering の寄与は Lommel-Seeliger term ( $S_{LS}$ )、multiple scattering の寄与は Lambert term ( $S_L$ ) で表される。

$$S_{LS} = \frac{\mu\mu_0}{\mu + \mu_0}, \quad (1.3)$$

$$S_L = \mu\mu_0. \quad (1.4)$$

(それぞれの物理的意味は?)

結論として結果に大きな影響を与えないでユーザは 0.1 というデフォルト値に固定することが多い (DAMIT ドキュメント)。小惑星のアルベドやレゴリス粒径の情報があれば使うべきにも感じるが、結果は変わらないのであるか。

つまり、LI では以下の表式の散乱特性を用いる。

$$S(\mu, \mu_0, \alpha) = f(\alpha)[S_{LS}\mu, \mu_0] + cS_L(\mu, \mu_0)], \quad (1.5)$$

ただし  $f(\alpha)$  は次節で述べる位相関数 (phase function) である。

### 1.4.2 Phase function

DAMIT で実装されている位相関数は以下の通り (e.g., Kaasalainen+2001, Hanus+2021)。

$$f(\alpha) = A_0 \exp\left(\frac{-\alpha}{D}\right) - k\alpha + 1 \quad (1.6)$$

ただし  $\alpha$  は位相角、 $A_0$  は小さい位相角での衝効果 (opposition effect) の強さを表す定数、 $D$  は衝効果の形状を表す定数、 $k$  は大きい位相角での位相曲線の傾きを表す定数。パラメタのデフォルト値は  $(A_0, D, k) = (0.5, 0.1, -0.5)$ 。これらをフリーパラメタにするか否かは DAMIT 内で指定することができるが、DAMIT ドキュメントによると結果に影響を与えないで固定することが多い。Hanus+2021 のようにサーベイ観測のスペースなデータを用いて LI をする際には相対等級ではなく絶対等級ベースで解析を行う。その際にはどうやらこれらを free parameters にするらしい。(等級単位の位相曲線との関連)

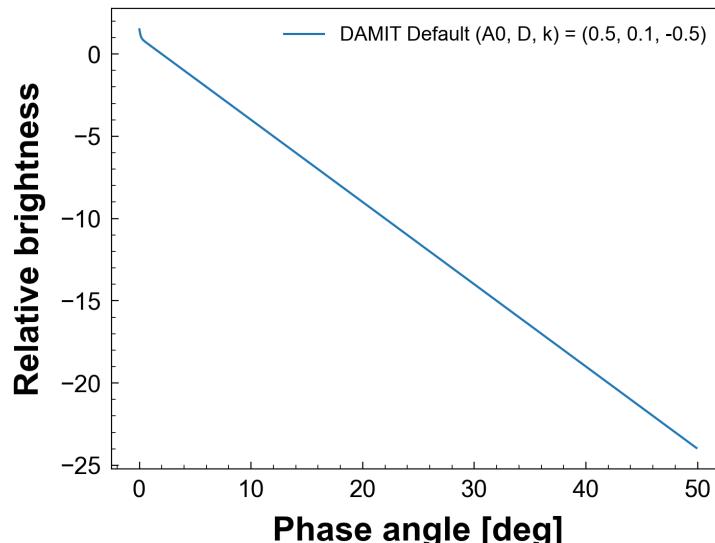


図 1.3 DAMIT で実装されているデフォルトの位相関数

## 1.5 計算

convex inverse problem (多面体逆問題) は以下の表式で記述される。

$$\vec{L} - A\vec{g} \quad (1.7)$$

ただし  $\vec{L}$  は観測された明るさを表すベクトル、 $\vec{g}$  は求めたいパラメタを表すベクトル。 $\vec{g}$  は curvature function を記述し (Gaussian surface density)、ユニークに形状を決定する。 $g$  の中のパラメタは convex polyhedron(凸多面体) の

## 1.6 まとめ

# 第2章

## 準備

本稿で用いる全てのソフトウェアのインストール手順を説明する。筆者の環境 MacBookPro (2021、M1、macOS Monterey 12.2) での説明であるため、他の環境では不都合が生じる可能性があることをご了承いただきたい。(2022-05-29、Linux Mint 19.2(Tina) でも問題なくコンパイルに成功した。) c、fortran、python 環境を用意する必要がある。必要な python モジュールは後述する自作モジュール `smfl` のスクリプトを確認していただきたい。

### 2.1 ソフトウェアのインストール

#### 2.1.1 DAMIT (Database of Asteroid Models from Inversion Techniques)

DAMIT のウェブサイト (<https://astro.troja.mff.cuni.cz/projects/damit/>) 上部の More タブ、Software download から Source codes v0.2.1 (tar-gzip 325 KiB) を押し圧縮ファイルをダウンロードする。以下コマンドで解凍する。

Listing 2.1 解凍

```
1 tar -zxvf version_0.2.1.tar.gz
```

`version_0.2.1` が生成され、その中に `period_scan`、`convexinv` など後ほど用いるプログラムが入っている。

#### 2.1.2 smfl (Shape Modeling From Lightcurves)

以下コマンドを実行して `smfl` (shape modeling from lightcurves) をダウンロードする。

Listing 2.2 smfl インストール

```
1 git clone https://jin_beniyama@bitbucket.org/jin_beniyama/smfl.git
```

`smfl`([https://bitbucket.org/jin\\_beniyama/smfl/src/master/](https://bitbucket.org/jin_beniyama/smfl/src/master/)) は DAMIT コードを効率的かつユーザフレンドリーに取り扱うための Python ラッパーである。大別するとデータ整形用スクリプト、データ可視化スクリプトに分けられる。`smfl` は bitbucket 上で管理されている。以降、`smfl` にパスが通っていることを前提とする。

#### 2.1.3 Metasequoia

3D モデリングソフトウェア Metasequoia をウェブサイト <https://www.metaseq.net/jp/> からダウンロードする。Metasequoia を用いることで小惑星の形状推定結果を含む論文でしばしば目にする「灰色の多面体の小惑星」を描くことができる。ただし本稿で行う stl (Standard Triangulated Language) フォーマットの可視化は有料 (EX) 版でしか行えない。試用期間が 30 日あるのでひとまず感覚をつかむという目的は達成可能である。論文を書くとなると課金する必要がある。

この後にレンダリング用ソフトウェア Persistence of Vision Raytracer (POV-Ray) を用いるべきかもしれない。他にも 3D モデリングソフトはあると思われるため、おすすめソフトがあれば教えていただきたい。

## 2.2 コンパイル

以下コマンドを機械的に実行することで DAMIT コード群をコンパイルする。まずは自転周期決定コード `period_scan`、形状推定コード `convexinv` をコンパイルする。

---

Listing 2.3 version\_0.2.1/convexinv/にて

---

```
1 make
```

---

膨大なライトカーブや高速観測データを用いる場合、コンパイル前に `constants.h` の該当項目を編集しておかないとコードが動かない。筆者は `POINTS_MAX=3000`、`MAX_N_OBS=30000` とした。この組み合わせによってはエラーがでるため要注意。原因は不明。

次にライトカーブ生成コード `lcgenerator` がコンパイルする。ただし筆者は使っていない。

---

Listing 2.4 version\_0.2.1/lcgenerator/にて

---

```
1 make
```

---

より高精度な形状推定コード (?)`conjgradinv` をコンパイルする。

---

Listing 2.5 version\_0.2.1/conjgradinv/にて

---

```
1 make
```

---

なお Mac ユーザはデフォルトの `clang` では c 言語スクリプトをコンパイルできない可能性がある。その場合は以下コマンドで `homebrew` から `gcc` をダウンロードする。

---

Listing 2.6 gcc インストール

---

```
1 brew install gcc
```

---

その後ダウンロードした `gcc` にパスを通す。筆者は `.zprofile` に `gcc` のエイリアスを作成した。以下コマンドを実行して `homebrew` からダウンロードした `gcc` が動いていれば問題ない。

---

Listing 2.7 gcc の確認

---

```
1 gcc -v
```

---

以上で必要ソフトのインストールとコンパイルは完了し、「灰色の多面体の小惑星」を描く準備が整った。

# 第3章

## 自転周期の決定

この章では観測で得たライトカーブ（時間、明るさ）を DAMIT コード用フォーマットに整形し、`period_scan` を用いて自転周期を決定する。複数日の観測データでも繋ぎ合わせて自転周期を推定することができる。推定した自転周期は次章以降の入力となる。主な出力は自転周期 vs. 適合度 ( $\chi^2$ ) である。他の手法で自転周期を精度良く決定できているのであればこの章を省略することができるが、個人的には `period_scan` を用いた自転周期推定も実施して周期推定結果の整合性を確かめることをお薦めする。

### 3.1 準備

必要なファイルは時間 (julian day) と明るさ (以下フラックスと呼ぶ) を含むテキストデータ (ライトカーブ) のみである。前処理として以下を実行することで時間、フラックスに加えて太陽、地球からの相対位置を含んだテキストファイルを得る。以下この処理を行った後のライトカーブを `convexinv` 形式のライトカーブと呼ぶ。

Listing 3.1 前処理 (period\_scan 用)

---

```
1 format4convexinv.py (inp_lc) --jpl (jp1) --N_mc (N) --obj (obj) --out (out_lc)
```

---

ただし `inp_lc` は入力ライトカーブ、`jp1` はライトカーブの時刻を含む該当天体の天体暦、`N_mc` はモンテカルロ法の試行回数、`obj` は天体名、`out_lc` は出力ファイル名である。`N_mc` を 2 以上に設定することでランダムにサイコロをふり複数の擬似ライトカーブを作ることができる。この際入力ファイルはフラックスの不定性を含む必要があり、フラックスはフラックスの不定性を標準偏差とする正規分布に従うと仮定してリサンプリングされる。

以降、DAMIT コードに含まれる小惑星 (43)Ariadne を例に進める。Ariadne のライトカーブ `test_lcs_rel` はすでに `convexinv` 形式なので、上記の前処理は必要ない。`convexinv` 形式のライトカーブは以下のコマンドで描写できる (図 3.1)。`rotP` オプション (単位は h) を与えると自転周期で折り返された光度曲線が得られる (図 3.1)。ここでは JPL/HORIZONS の値 5.76218 h を採用する。なおデフォルトでは自転周期の単位は秒なので、時間単位で与えるには `hour` オプションを付与する必要がある。現在はライトカーブの数が 8 個以上の時は最初の 8 個のみが描写される。Ariadne の場合は 37 個中の最初の 8 個である。

Listing 3.2 ライトカーブの確認

---

```
1 plot_lcs_with_model.py Ariadne test_lcs_rel --rotP 5.76218 --hour
```

---

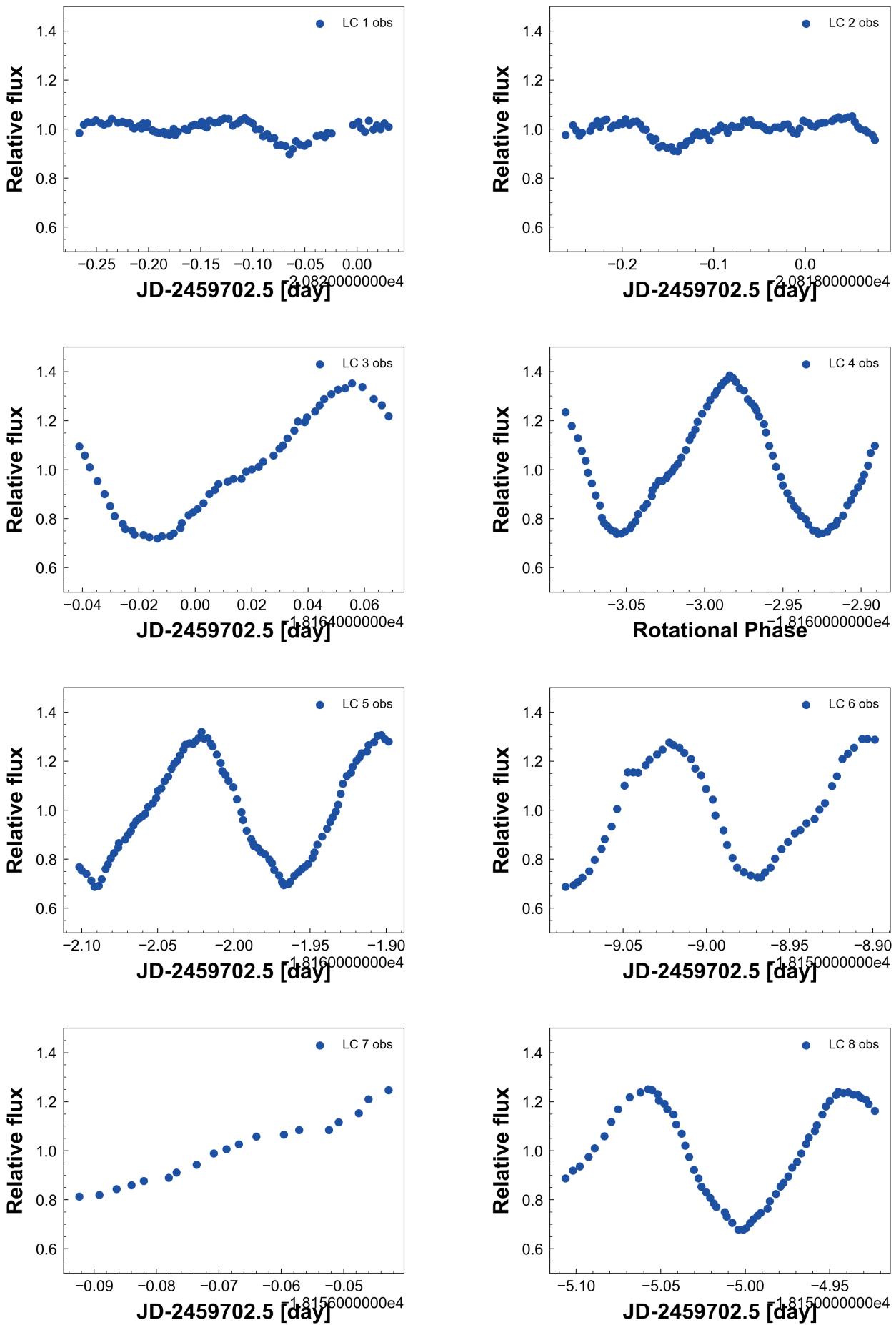


図 3.1 Ariadne のライトカーブ

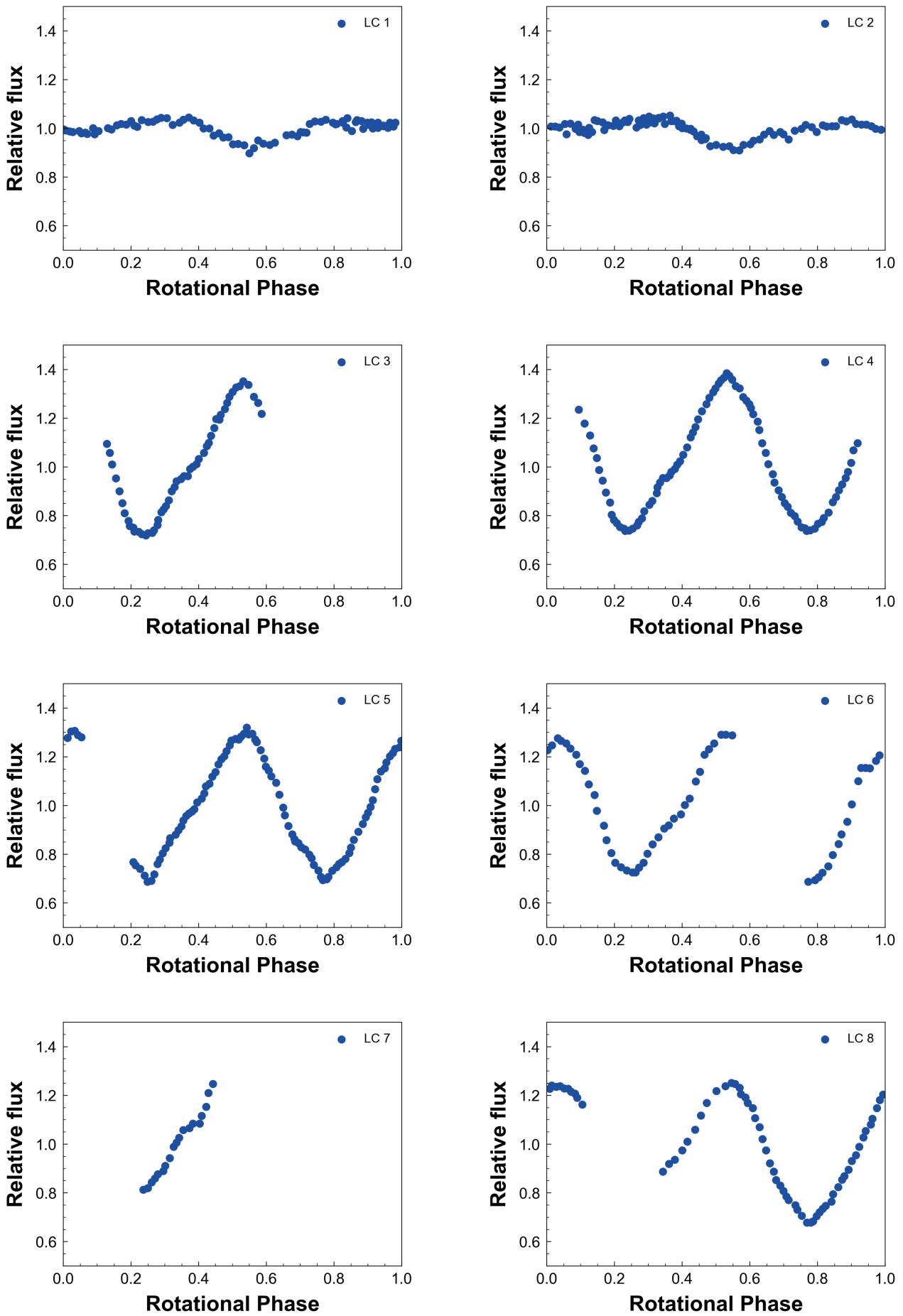


図 3.2 自転周期で折り返した Ariadne のライトカーブ

次に自転周期推定のための入力ファイルを用意する。Ariadne の自転周期推定のための入力ファイル `input_period_scan` は以下の通り。

Listing 3.3 `period_scan` の入力 (Ariadne の例)

---

```

1 5.7615 5.7625 0.8 period start - end - interval coeff.
2 0.1    convexity weight
3 6 6    degree and order of spherical harmonics
4 8     no. of rows
5 0.5 0  scattering parameters
6 0.1 0
7 -0.5 0
8 0.1 0
9 50    iteration stop condition
10 10   minimum number of iterations (only if the above value < 1)

```

---

各数値の後に説明が付与されている。1 行目は自転周期探索範囲の最小値、最大値、探索幅に対応する係数。

$$\frac{\Delta P}{P} = \frac{1}{2} \frac{P}{T} \quad (3.1)$$

という式に基づいて周期のグリッドが決まる。よってグリッド数  $N_{\text{grid}}$  は

$$N_{\text{grid}} = \frac{2\Delta t(P_1 - P_0)}{P_0 P_1} \frac{1}{p} \sim \frac{(P_1 - P_0)}{\Delta P} \frac{1}{p} \quad (3.2)$$

で求まる。定数  $p$  を小さくすれば細かいグリッドとなる。

2 行目は形状の重み (?)。3 行目、4 行目は形状の複雑さを表す数。5 行目は位相関数を表す定数。6 行目は? 7 行目は? 8 行目は? 9 行目は最大の試行回数で、それでも収束しないと形状推定が終了する。10 行目は最小試行回数で、この回数はイテレーションが回る。詳細な解説は DAMIT コードの中にある pdf ファイルを参照のこと。

## 3.2 実行と描写

Ariadne のテストデータは複数のライトカーブが一つのテキストファイルにまとめられている。以下で `period_scan` を実行する。ライトカーブが多いとそれなりに時間がかかる (未評価)。

Listing 3.4 単一ライトカーブの周期推定 (Ariadne の例)

---

```
1 search_sidP.py test_lcs_rel --inp input_period_scan --out out_period_scan
```

---

なお複数ライトカーブを引数にとることも可能である。ただし個人的には前処理でまとめておく方が良いと思う。

Listing 3.5 複数ライトカーブの周期推定

---

```
1 search_sidP.py (lc1) (lc2) --inp (inputfile) --out (out)
```

---

周期推定結果は以下で描写する。

Listing 3.6 単一ライトカーブの周期推定結果の描写

---

```
1 plot_sidP_chi2.py Ariadne out_period_scan
```

---

図 3.3 より、自転周期は適合度 ( $\chi^2$ ) が最小をとる 5.76198h と推定される。不定性を評価するためには次章の MC 法を用いる。

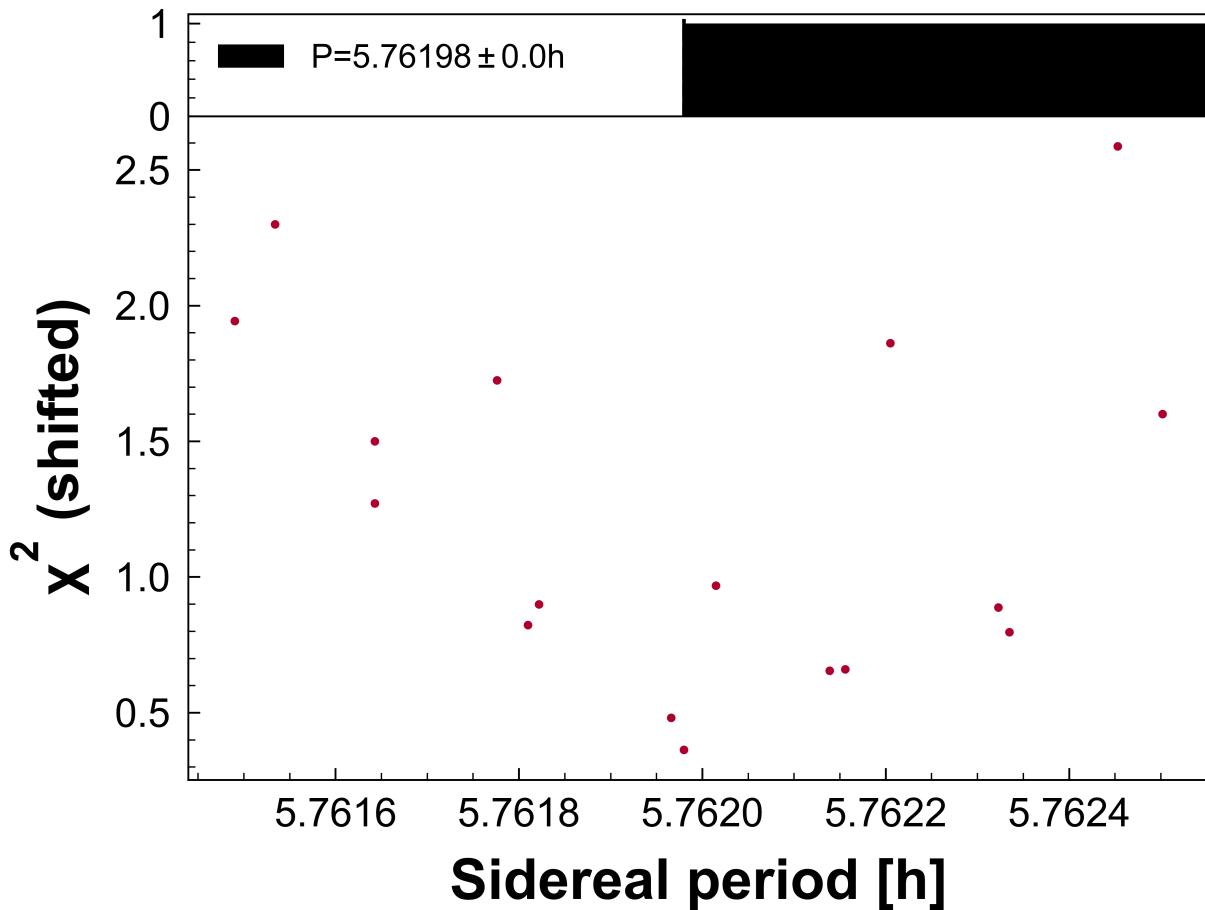


図 3.3 Ariadne の period\_scan 結果。時間短縮のためグリッド数を少なくしたので見栄えが悪い。

### 3.3 実行と描写 (MC 法)

$N_{mc}$  を 2 以上にして複数のライトカーブを作成してそれぞれに対して search\_sidP.py を実行すれば良い。ただしそのためには測光誤差の情報が必要で、DAMIT で配布されている test\_lcs\_rel はそれを含まない。複数のライトカーブの周期推定結果は以下で描写する。

Listing 3.7 複数ライトカーブの周期推定結果の描写

---

```
1 plot_sidP_chi2.py (obj) (lc1) (lc2) (lc3)
```

---

Ariadne ではない小惑星の例を図 3.4 に示す。各ライトカーブで最も  $\chi^2$  が小さい自転周期の標準偏差を不定性として計算する。

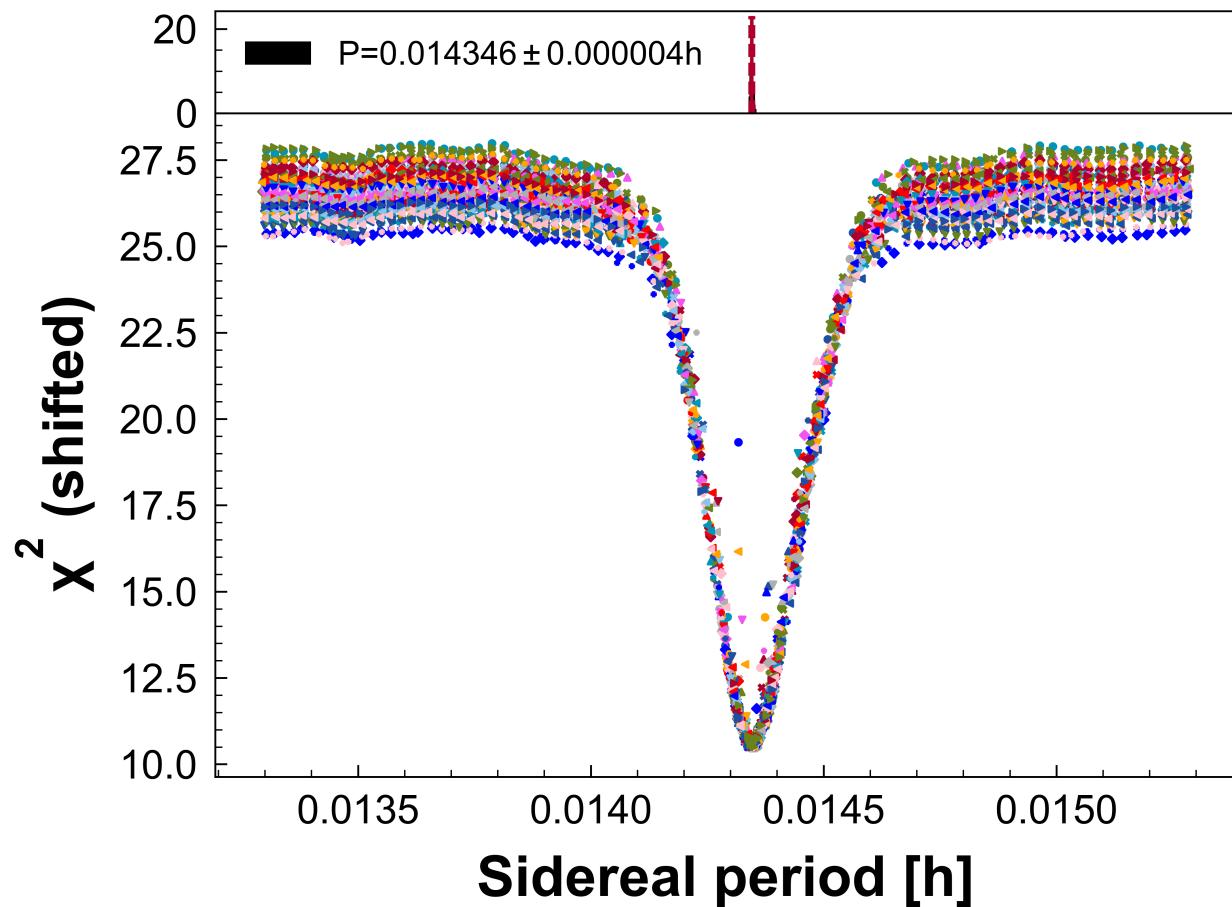


図 3.4 とある小惑星の period\_scan 結果。各色が一つのライトカーブに対する結果を表す。

## 第 4 章

# 自転軸の推定、形状モデルの作成

この章では前章で推定した自転周期を初期値として自転する小惑星を考える。自転軸の向き、形状、自転周期をフリーパラメタにして `convexinv` を用いた形状推定を実行する。主な出力は自転軸北極の黄経、黄緯のフィット度合い ( $\chi^2$ ) のマップである。この結果推定した自転軸、自転周期を固定して形状モデルを作成し、可視化する。さらに得られた形状に太陽光が当たった際に得られるモデルカーブと観測データを比較する。

### 4.1 軸推定

まずは以下の前処理を実行して代表的な光度曲線を取得する。必要なファイルは時間 (julian day)、フラックスを含むテキストデータのみである。Section 3.1 と同様に前処理として以下を実行することで時間、フラックスに加えて太陽、地球からの相対位置を含んだテキストファイルを得ることができる。

Listing 4.1 前処理 (軸推定用)

---

```
1 format4convexinv.py (lc1) (lc2) (lc3) --jp1 (jp11) (jp12) (jp13) --obj (obj) --out (
    lcs_all)
```

---

Section 3.1 とは異なり  $N_{mc}$  は 1 に設定する。形状はモンテカルロ法を用いずに一意に決定する。なお前章でも述べたとおり Ariadne のテストデータはすでに補正されているのでこの処理は行わなくてよい。

以下を実行することで任意の数の自転軸の方向に対応した `convexinv` の入力ファイルが作成できる。特別な理由がない限り `fixpole` オプションをつけて自転軸を固定する。自転周期は固定しない。以下コマンドで黄経  $\lambda$ 、黄緯  $\beta$  をともに 30 分割した 900 の軸に対して形状モデルを作成する。作成した入力ファイルは `convexinv_input` に保存される。

Listing 4.2 input ファイルの作成

---

```
1 make_convexinv_input.py --sidP 5.76218 --Nlam 30 --Nbta 30 --fixpole
```

---

以下は  $(\lambda, \beta) = (198^\circ, 90^\circ)$  の例。

Listing 4.3 convex\_inv の入力例 (inp\_ci\_198\_90)

---

```
1 198.6206896551724 0
2 90.0 0
3 5.76218 1
4 0
5 0
6 0.1
7 6 6
8 8
9 0.5 0
10 0.1 0
11 -0.5 0
12 0.1 0
13 50
```

---

この入力ファイルを用いて convexinv を実行する。出力ファイルは convexinv\_result に保存される。

Listing 4.4 convexinv の実行

---

```
1 do_convexinv.py --Nlam 30 --Nbta 30 --lc ../test_lcs_rel --lcdir .
```

---

以下は  $(\lambda, \beta) = (198^\circ, 90^\circ)$  の例。

Listing 4.5 convex\_inv の出力例 (res\_ci\_198\_90)

---

```
1 198.621 90 initial lambda, beta (0,0)
2 5.762180 initial period (hrs) (1)
3 0 epoch of zero time t0
4 0 initial fixed rotation angle fi0
5 0.1 the weight factor for conv. reg.
6 6 6 degree and order of the Laplace series
7 8 nr. of triangulation rows per octant
8 0.5 0.1 -0.5 initial guesses for phase funct. params. (0,0,0)
9 0.1 initial Lambert coeff. (Lommel-Seeliger part = 1) (0)
10 50 stop condition
11
12 the used epoch of zero time 2438882.000000
13
14 1 chi2 21.527167 dev 0.118193 alambda 0.010000
15 2 chi2 21.527167 dev 0.118193 alambda 0.100000
16 3 chi2 21.527167 dev 0.118193 alambda 1.000000
17 4 chi2 21.527167 dev 0.118193 alambda 10.000000
18 5 chi2 21.527167 dev 0.118193 alambda 100.000000
19 6 chi2 20.510534 dev 0.115368 alambda 10.000000
20 7 chi2 16.183482 dev 0.102479 alambda 1.000000
21 8 chi2 12.491251 dev 0.090033 alambda 0.100000
22 9 chi2 12.491251 dev 0.090033 alambda 1.000000
23 10 chi2 12.275460 dev 0.089252 alambda 0.100000
24 11 chi2 12.275460 dev 0.089252 alambda 1.000000
25 12 chi2 12.135193 dev 0.088741 alambda 0.100000
26 13 chi2 12.135193 dev 0.088741 alambda 1.000000
27 14 chi2 12.012298 dev 0.088290 alambda 0.100000
28 15 chi2 12.012298 dev 0.088290 alambda 1.000000
29 16 chi2 11.894925 dev 0.087858 alambda 0.100000
30 17 chi2 11.894925 dev 0.087858 alambda 1.000000
31 18 chi2 11.777648 dev 0.087423 alambda 0.100000
32 19 chi2 11.777648 dev 0.087423 alambda 1.000000
33 20 chi2 11.670910 dev 0.087026 alambda 0.100000
34 21 chi2 11.670910 dev 0.087026 alambda 1.000000
35 22 chi2 11.571762 dev 0.086656 alambda 0.100000
36 23 chi2 11.316760 dev 0.085696 alambda 0.010000
37 24 chi2 11.316760 dev 0.085696 alambda 0.100000
38 25 chi2 11.117657 dev 0.084939 alambda 0.010000
39 26 chi2 11.117657 dev 0.084939 alambda 0.100000
40 27 chi2 11.051354 dev 0.084685 alambda 0.010000
41 28 chi2 11.051354 dev 0.084685 alambda 0.100000
42 29 chi2 11.020432 dev 0.084566 alambda 0.010000
43 30 chi2 11.020432 dev 0.084566 alambda 0.100000
44 31 chi2 10.995483 dev 0.084471 alambda 0.010000
45 32 chi2 10.995483 dev 0.084471 alambda 0.100000
46 33 chi2 10.969284 dev 0.084370 alambda 0.010000
47 34 chi2 10.969284 dev 0.084370 alambda 0.100000
48 35 chi2 10.947575 dev 0.084286 alambda 0.010000
```

---

```
49 36 chi2 10.947575 dev 0.084286 alambda 0.100000
50 37 chi2 10.925095 dev 0.084200 alambda 0.010000
51 38 chi2 10.925095 dev 0.084200 alambda 0.100000
52 39 chi2 10.901066 dev 0.084107 alambda 0.010000
53 40 chi2 10.901066 dev 0.084107 alambda 0.100000
54 41 chi2 10.881577 dev 0.084032 alambda 0.010000
55 42 chi2 10.881577 dev 0.084032 alambda 0.100000
56 43 chi2 10.863944 dev 0.083964 alambda 0.010000
57 44 chi2 10.863944 dev 0.083964 alambda 0.100000
58 45 chi2 10.845124 dev 0.083891 alambda 0.010000
59 46 chi2 10.845124 dev 0.083891 alambda 0.100000
60 47 chi2 10.826998 dev 0.083821 alambda 0.010000
61 48 chi2 10.826998 dev 0.083821 alambda 0.100000
62 49 chi2 10.811131 dev 0.083760 alambda 0.010000
63 50 chi2 10.811131 dev 0.083760 alambda 0.100000
64
65 lambda, beta and period (hrs): 198.620690 +90.000000 5.762168
66 phase function parameters: 0.500000 0.100000 -0.500000
67 Lambert coefficient: 0.1
68 plus a dark facet with area 0.72%
```

---

以下で結果を描写する(図4.1)。デフォルトではconvexinv\_resultに保存されている出力ファイルが参照されるので、基本的には引数は2つのみで良い。

Listing 4.6 軸方向結果の描写

---

```
1 plot_polesolution.py --Nlam 30 --Nbta 30
```

---

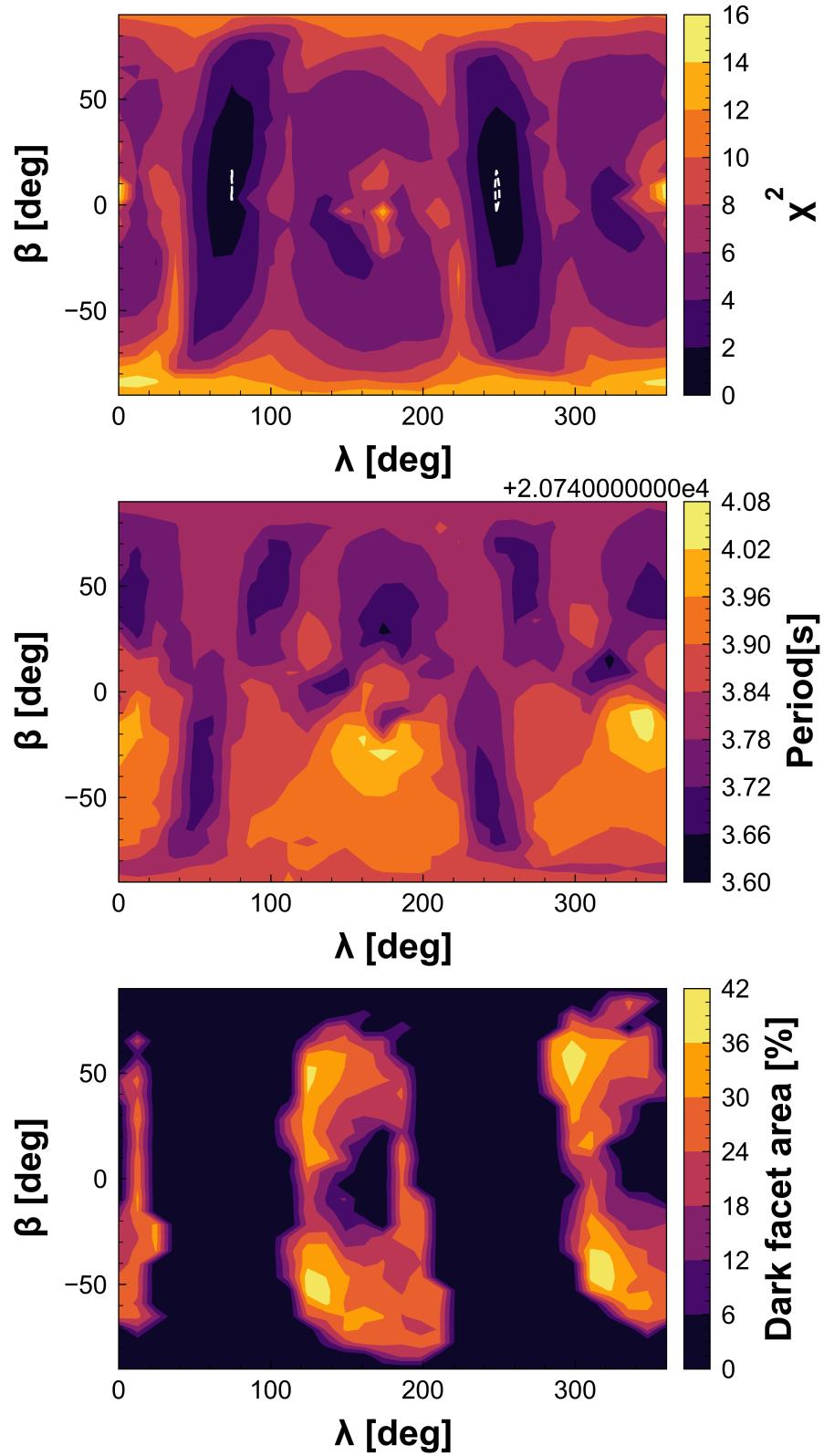


図 4.1 Ariadne の自転軸推定結果。作成した形状モデルのモデルカーブと観測の適合度 ( $\chi^2$ ) がコントアで表されている。 $(\lambda, \beta = 80^\circ, 0^\circ)$ 、 $(\lambda, \beta = 250^\circ, 0^\circ)$  付近の白い領域は 3-sigma 信頼区間（詳細は本文を参照）。

この自転軸推定マップを用いて自転軸の方向を決定し、形状モデルを作成したいが、その解の判断は慎重に行う必要がある。マップの中で最も低い  $\chi^2$  をもつ黄経  $\lambda$  と  $\beta$  の組み合わせが解の候補であることは間違いないが、 $\chi^2$  が低いだけでは十分条件を満たさない。先行研究で広く用いられている統計的手法に基づいて解が信頼できるか否かを判断する。同様の計算は Vokrouhlicky2011、Vokrouhlicky2013、Durech2018、Hanus2018 でも行われている。ただし、Hanus2018 以外の論文は reduced  $\chi^2$  ではなく通常の  $\chi^2$  を用いた式になっていることに注意が必要である。すなわち、以下の最終表式は Hanus2018 と一致するが、それ以外とは一致しない。

まず… 信頼区間は以下のように計算する。(in prep.)

## 4.2 最終形状モデル推定の準備

さて、ここまでで自転周期と自転軸の向きが決定された。DAMIT によると縮退している解のうち Ariadne の自転軸は  $(\lambda, \beta = 253^\circ, -15^\circ)$  であるらしい。入力ファイルは以下で作成する。

Listing 4.7 input ファイルの作成 (周期、軸固定)

```
1 make_convexinv_input.py --sidP 5.76218 --lam 253 --beta -15 --fixpole
```

自転周期、自転軸を固定した時の形状モデルを `convexinv` を用いて作成する。出力ファイルは `convexinv_final` に保存される。

Listing 4.8 input ファイルの作成 (周期、軸固定)

```
1 do_convexinv.py --lc ./test_lcs_rel --lam 253 --beta -15 --final
```

## 4.3 形状モデルとモデルカーブの可視化

Metasequoia で可視化するために `convexinv` の出力モデルを `stl` 形式に変換する。

Listing 4.9 stl へ変換

```
1 model2stl.py model_ci_253_-15
```

以下コマンドを実行することで「灰色の多面体の小惑星」が出力される(図 4.2)。ファイル、レンダリングから画像を保存する。ただし Metasequoia は 1 ヶ月経つとシリアルコードの有効期限が切れてしまうことに注意。

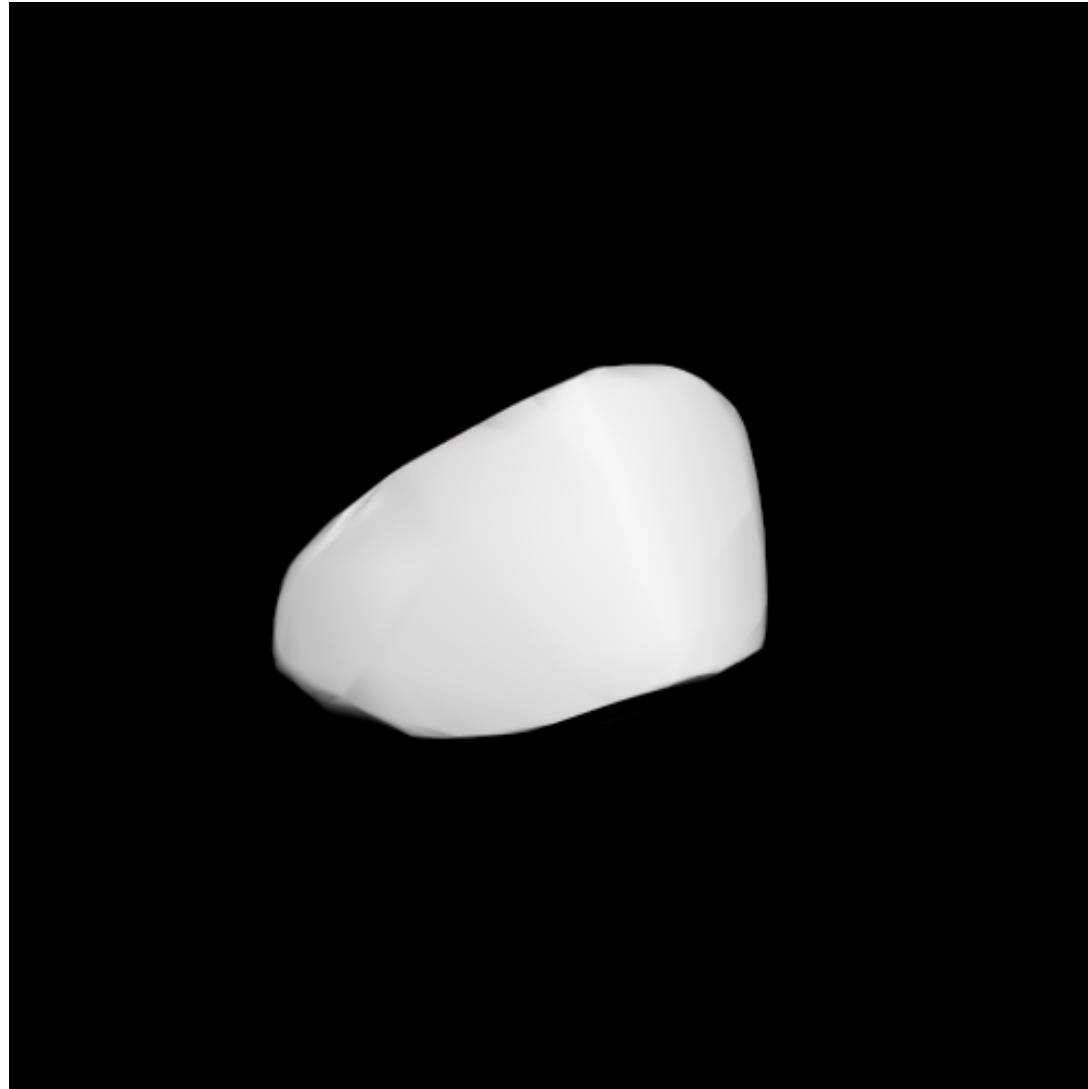


図 4.2 Ariadne の形状モデル

以下でモデルカーブ(赤)、観測データ(青)を描写することができる(図4.3)。自転周期を引数として与えることで折り返し光度曲線を描写することもできる(図4.3)。ただし、現状のコードでは先頭の8つのライトカーブのみが描写される。

Listing 4.10 観測とモデルの比較

---

```
1 plot_lcs_with_model.py Ariadne ../test_lcs_rel --rotP 5.76218 --lc_model convex_final/
  outlcs_ci_253_-15 --hour
```

---

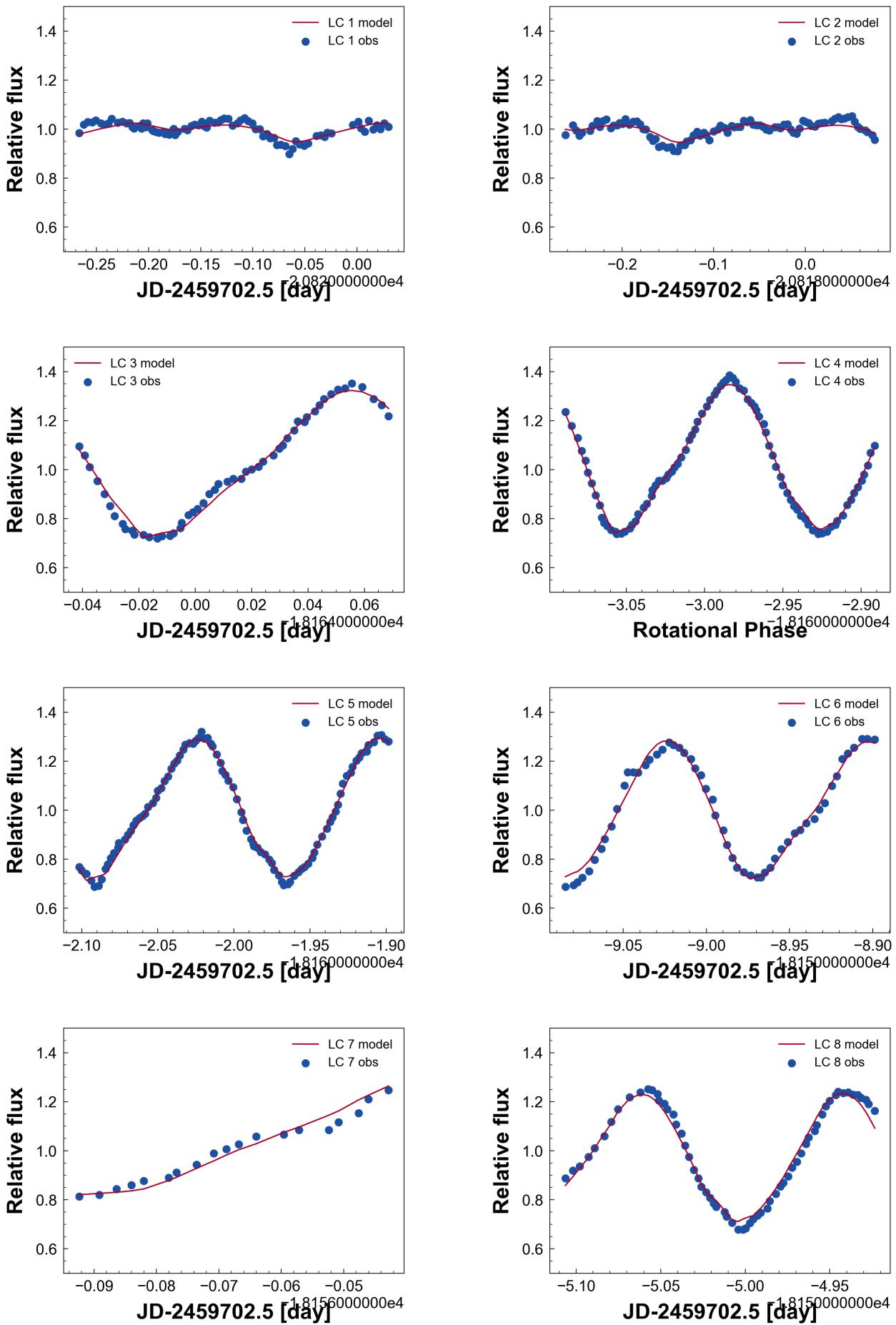


図 4.3 Ariadne の観測結果とモデルカーブ

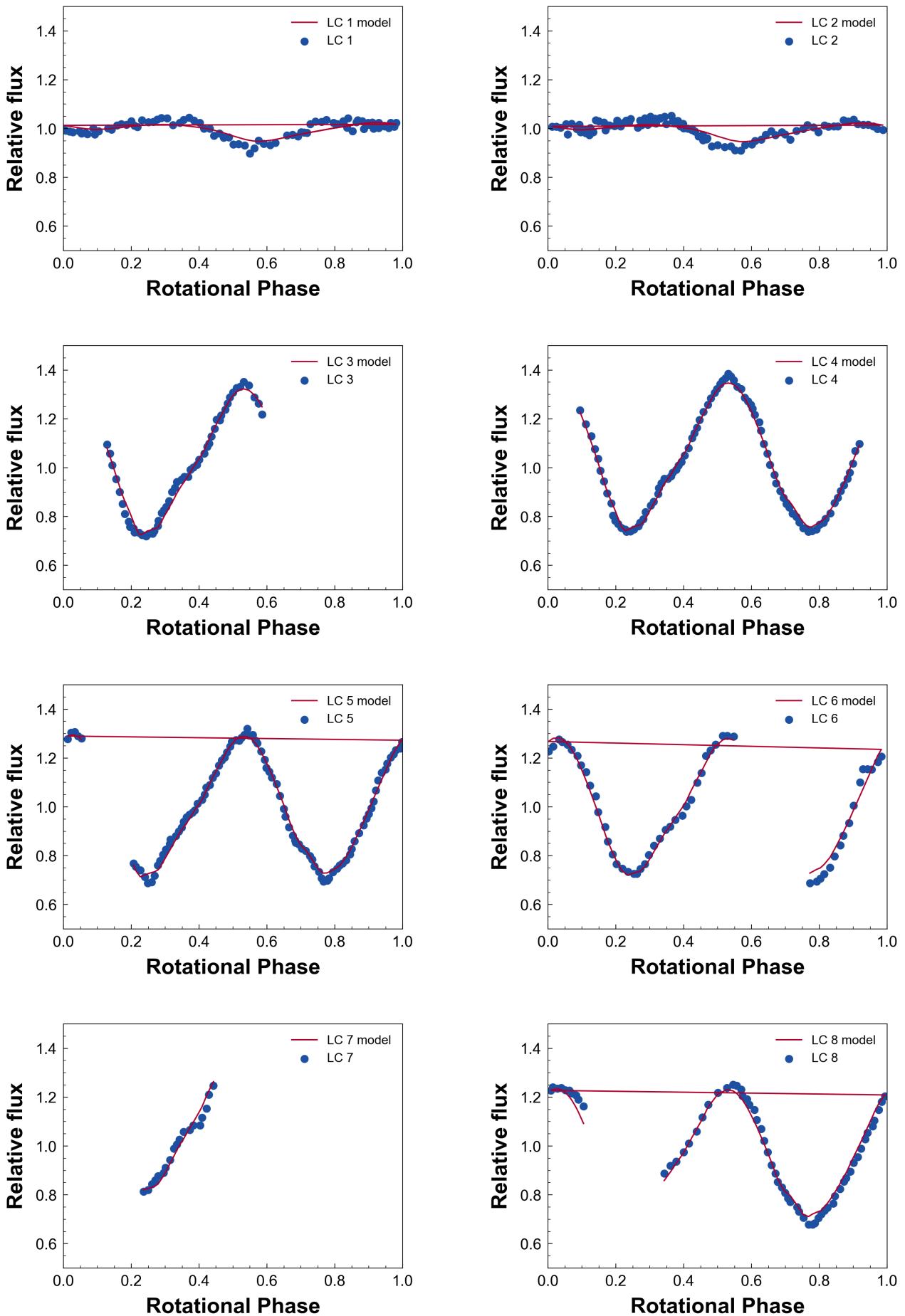


図 4.4 自転周期で折り返した Ariadne の観測結果とモデルカーブ