# 放電加工機の時系列解析（その３）

## 1 はじめに

### 1.1 時系列の取組について

放電加工機の時系列解析に取り組みます。

時系列解析は、以下の４ステップから構成されます。

① ディスクリプション (description)

時系列の特徴を把握します。定常性の確認、成分分解等を行います。

② **モデリング（modeling）**

時系列モデルを構成し、パラメータを推定します。

③ **予測 (prdiction)**

現在までの情報から今後の変動を予測します。

④ 信号抽出

必要な信号や情報を取り出す。(異常検知とか)

今回の報告書は、上記の **モデリング（modeling）**と **予測 (prdiction)** についての取組内容です。モデルについて、代表的な手法として、自己回帰モデル と **状態空間モデル**がありますが、今回は、**状態空間モデル**についての報告です。

### 1.2 どのようなデータを扱ったか

放電加工機は、加工中に障害が発生すると自己診断して 3 段階のレベルに分け、レベルに応じたメッセージを表示します。内容は以下です。

・エラーメッセージ：続行不可能な障害が発生したときに表示し、動作を中断する。

・ハルトメッセージ：再開可能な障害が発生したときに表示し、一時停止する。

・コメントメッセージ：続行可能な障害で発生し、注意を促す。

また、メッセージの記録は、USB により CSV データとして取り出すことが可能です。

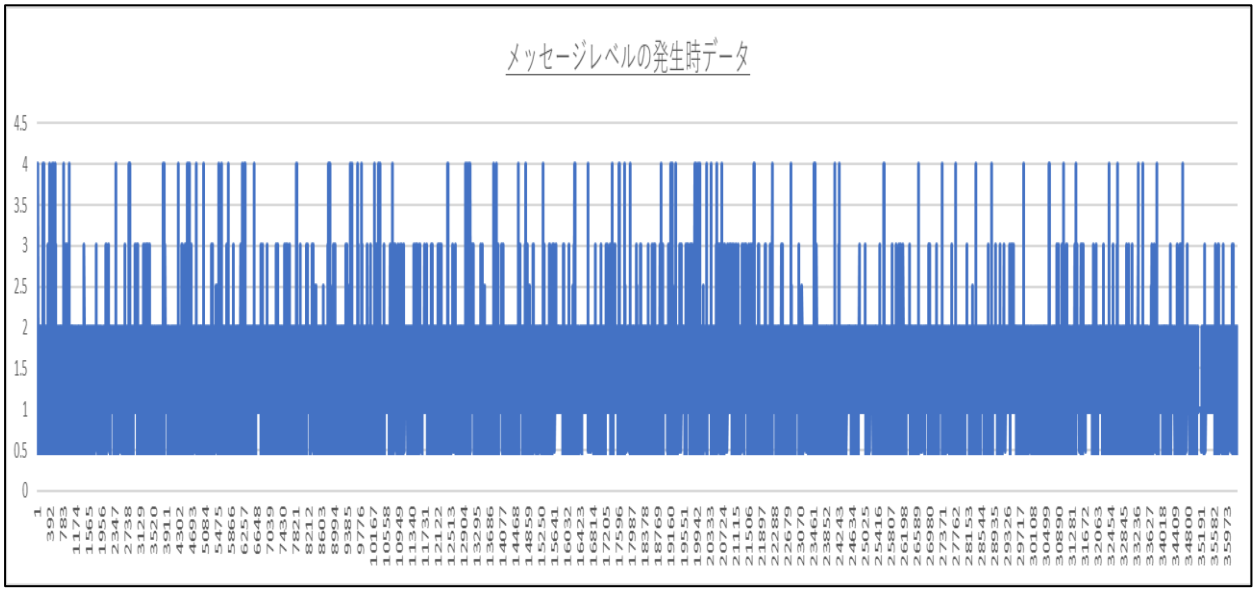
レベルに対応してデータ処理できるように、3 段階のレベルについて障害のレベルが大きいと、点数は大きくなるように点数を設けました。

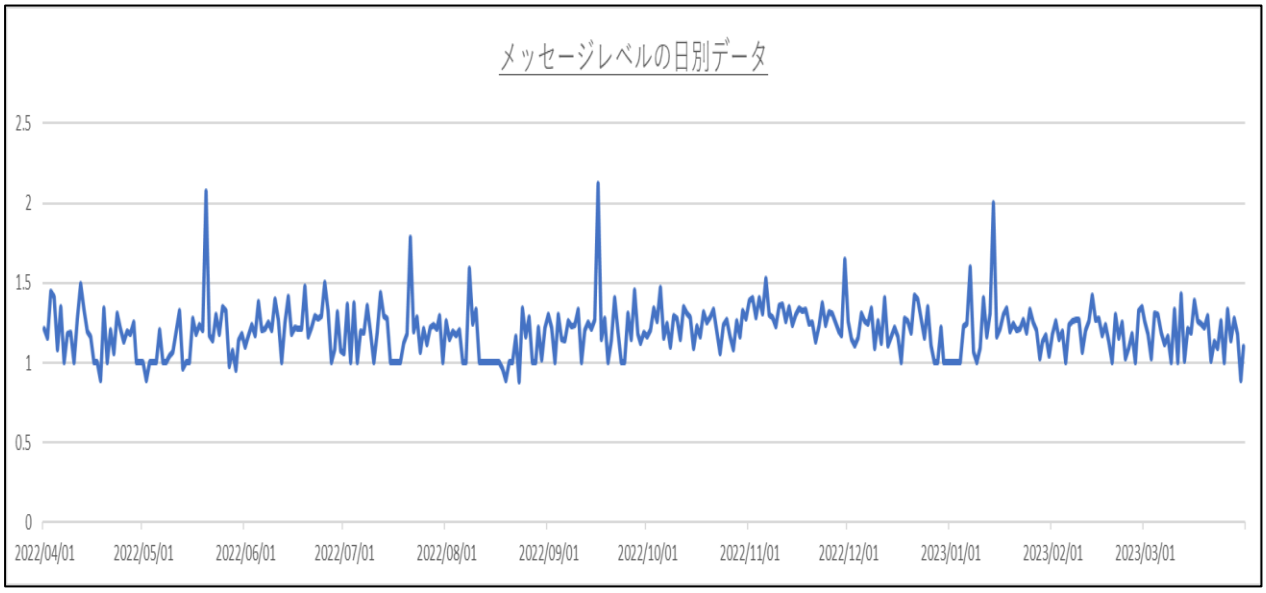
これらは、メッセージが発生した時の時間毎のデータで、点過程データと呼ばれるもので、

時系列データではありません。故に、一日分の平均値を取って時系列データとして取り出すようしました。本データを以下に示します。2022 年 4 月　から　2023 年 3 月までのデータです。

　上段が発生時毎のデータ（点過程データ）で、1 年で 37000 回発生してます。1 点の発生は、一定時間の間隔ではありません。

　下段は点過程データを、一日毎に平均して、平均したデータです。点の間隔は 1 日毎ですので、時系列データと言えます。



## 1.3　プログラムについて

今回　以下三つのパッケージで計算して、予測しています。

①　R　の　時系列解析パッケージ TSSS　で計算します。

　　１変量 AR モデルのあてはめ　→　arfit

　　状態空間モデルによる時系列の予測と補間　→　tsmooth

　　https://jasp.ism.ac.jp/ism/TSSS/　を参照ください。

②　Python　の　statsmodels.tsa.statespace　で計算します。

　モデルのあてはめ　　→　UnobservedComponents()

　但し　ローカルレベルモデル（次章で説明します）に対応したものです。

③　Python　の　Pykalman　と　scipy.optimize の minimize 関数、で計算します。

　KalmanFilter　計算　→　Pykalman

　パラメータの推定　→　　scipy　は数値解析ソフトウェア・ライブラリーで

　　optimize　最適化を求めるツールで,minimize　は最小化を行います。

本報告書にはプログラムは記載していません。

～参考文献、WEB～

「R による時系列モデリング入門」　北川源四郎　著　（株）岩波書店

時系列解析　-自己回帰モデル･状態空間モデル･異常検知-　島田直希　著　共立出版（株）

第 3 回　時系列分析～状態空間モデル編～

　　https://note.com/yiida/n/n5e5db1bea552

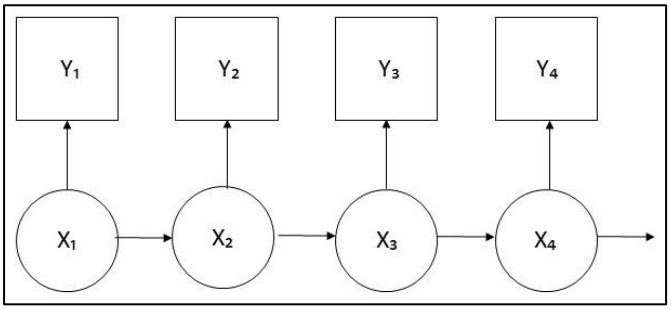
## 2 状態空間モデルについて

先に報告した自己回帰モデルによる解析は、観測値を直接モデル化しました。

状態空間モデルは、状態と観測値に分解してモデル化する手法で、単一のモデルというより、様々なモデル解析を統合したフレームワークといえます。

### 2.1 状態空間モデルとは

状態値と観測値の関連を図で表示しました。X が状態値で、Y が観測値となり、状態値は 1 時点前の状態に影響され、観測値は、その時点の状態値に影響されます。




さらに、数式では以下となります。

$y_t$ を l 変量の時系列として、以下を状態空間モデルと呼びます。

$$x_t = F_t x_{t-1} + G_t v_t \quad \text{(状態モデル)}$$

$$y_t = H_t x_t + w_t \quad \text{（観測モデル）}$$

$x_t$: 観測できない k 次元のベクトルで、状態と呼びます。

$v_t$: 状態ノイズで、平均ベクトル 0，分散共分散行列$Q_t$に従う m 次元の正規白色雑音

$y_t$: 観測可能な時系列データで、l 次元ベクトル

$w_t$: 観測ノイズで、平均ベクトル 0，分散共分散行列$R_t$に従う l 次元の正規白色雑音

$F_t$ $G_t$ $H_t$: それぞれ 係数行列(k x k),係数行列(k x m),観測行列(l x k) の行列です。

状態空間モデルは、非定常、非線形でもモデル化が可能とされています。

次に、状態空間モデルで Local Level モデルというものがあり、以下の数式となります。

$$x_t = x_{t-1} + v_t \quad \text{(状態モデル)}$$

$$y_t = x_t + w_t \quad \text{（観測モデル）}$$

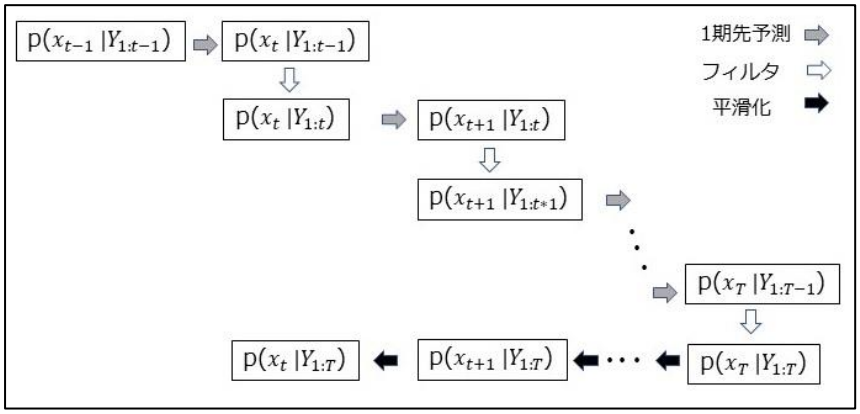
係数を 1 として、計算を簡略化しています。

（statsmodels.tsa.statespace に使用しているモデルです。）

## 2.2　状態の逐次推定とパラメータ推定

推定について述べますが、線形ガウス型モデルについての内容です。

状態の逐次推定は、以下の図で説明されます。




状態$x_t$ を推定することを状態推定と呼び、観測時点ｊ　と　ｔ の関係から、三つに分けます。

・1 期先予測 ： ｊ＜ｔ　・フィルタ ： ｊ＝ｔ　・平滑化 ： ｊ＞ｔ

$p(x_t \mid Y_{1:j})$　とは、時点１からｊまでの観測データ集合Ｙから、ｔ時のｘの確率分布を求めることを意味し、カルマンフィルタというアルゴリズムを用います。

また、$p(x_t \mid Y_{1:j})$　は正規分布に従うので、平均　$x_{t|j}$　と　分散共分散行列　　を求めることになります。

$x_{t|j}$　＝　$E[x_t \mid Y_{1:j}]$

$V_{t|j}$　＝　$E[(x_t - x_{t|j})(x_t - x_{t|j})^T \mid Y_{1:j}]$　　となり、

カルマンフィルタにより以下の計算式となります。

・一期先予測　　$x_{t|t-1} = F_t x_{t-1|t-1}$

$V_{t|t-1} = F_t V_{t-1|t-1} F_t^T + G_t Q_t G_t^T$

・フィルタ　　$K_t = V_{t|t-1} H_t^T (H_t V_{t|t-1} H_t^T + R_t)^T$

$x_{t|t} = x_{t|t-1} + K_t (y_t - H_t x_{t|t-1})$

$V_{t|t} = (I_k - K_t H_t) V_{t|t-1}$

Ｉ は単位行列

・平滑化　　$A_t = V_{t|t} F_{t+1}^T V_{n+1|n}^{-1}$

$x_{n|N} = x_{n|N} + A_n (x_{n+1|N} - x_{n+1|n})$

$$V_{n|N} = V_{n|n} + A_n \left(V_{n+1|N} - V_{n+1|n}\right)A_n^T$$

・長期予測　　i = 1,⋯,j について、

$$x_{n+i|n} = F_{n+1}x_{n+i-1|n}$$

$$V_{n+i|n} = F_{n+1}\,V_{n+i-1|n}F_{t+1}^T + G_{n+1}\,Q_{n+i}G_{t+1}^T$$

これらの計算式で、求めていくわけですが、F,G,H,Q を決めないと計算できません。

（仮に、想定した数列で計算を進めることもあります）

F,G,H,Q は、パラメータの推定により、 を計算していきます。

パラメータの推定について、先回の技術報告書に、数式として紹介しました。

今回も繰り返しになります。

パラメータは　θ　= $(a_1, \cdots\cdots\cdots a_m, \sigma^2)^T$　とします。

対数尤度は、　l(θ) = $\sum_{n=1}^{M} logp(\ y_n\ |y_1,\cdots,\ y_{n-1})$　となります。

さらに、l(θ)　を変換すると、　（参考文献を参照ください）

l(θ)　= -((N-M)/2)log2π$\sigma^2$　- 1/2$\sigma^2$ $\sum_{n=M+1}^{N}(\ y_n\ -\ \sum_{i=1}^{m} ai\, y_{n-i}\ )^2$

となり、l(θ) を　最大とする、パラメーターを算出します。

まずは、l(θ) を最大にする分散　$\sigma^2$　を求めると、（参考文献を参照ください）

$\sigma^2$　= $\frac{1}{N-M}\sum_{n=M+1}^{N}(\ y_n\ -\ \sum_{i=1}^{m} ai\, y_{n-i}\ )^2$　　となり

自己回帰係数　$a_1, \cdots\ a_m$　の対数尤度は

l($a_1, \cdots\ a_m$) =　-((N-M)/2)log2π$\sigma^2$　- (N-M)/2　となります。

分散　$\sigma^2$　　を最小化する　自己回帰係数を求めることになります。

今回は、

①Rの TSSS 解析パッケージにおいては、ユール-ウォーカー法、で計算しています。

②Python の statsmodels.tsa.statespace においては、そもそも、係数は１です。

③Python の　scipy.optimize の minimize 関数　においては、最適化手法として、SLSQP 法を、計算しています。

## 2.3　予測の結果

### 2.3.1　R 時系列解析パッケージ TSSS　での予測

観測モデルを AR モデルに従うとして、状態空間モデルを予測してます。

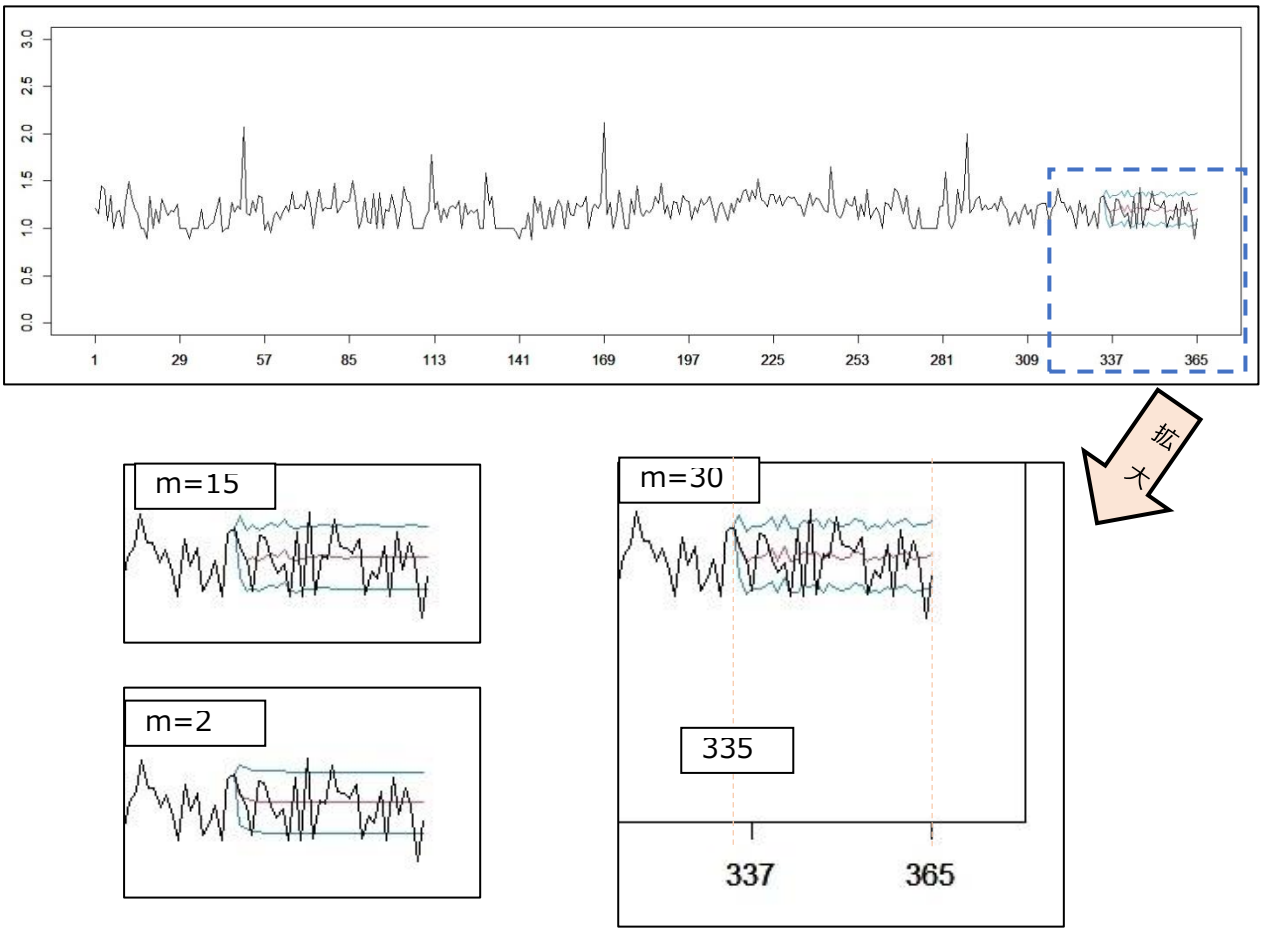
$y_n = \sum_{i=1}^{m} a_i \ y_{n-i} + v_n$　（観測モデル）　となり、

状態モデルを、$x_n = (y_n, y_{n-1}, \cdots y_{n-m+1})^T$　と定義すると、

$x_n = Fx_{n-1} + Gv_n$　という関係になり

$y_n = Hx_n$　で観測モデルが得られます。

本モデルは、状態モデルで、観測値で、完全に決定され、観測ノイズは 0 となるモデルと定義されています。この条件で、TSSS を用いて、計算しプロットしたグラフが以下です。出力されたグラフです。線を太くしたかったのですが、線太さの変更が R では難しく、デフォルトで出力してます。黒線が、観測値で、赤線が予測です（3 月分で x 軸で 335 から 365 の間)。また、青線の幅が、標準予測誤差区間です。



mは次数です。　m=2, m=15, m=30 と変化させてグラフを作成しました。

以下は、次数毎の対数尤度と AIC の値です。

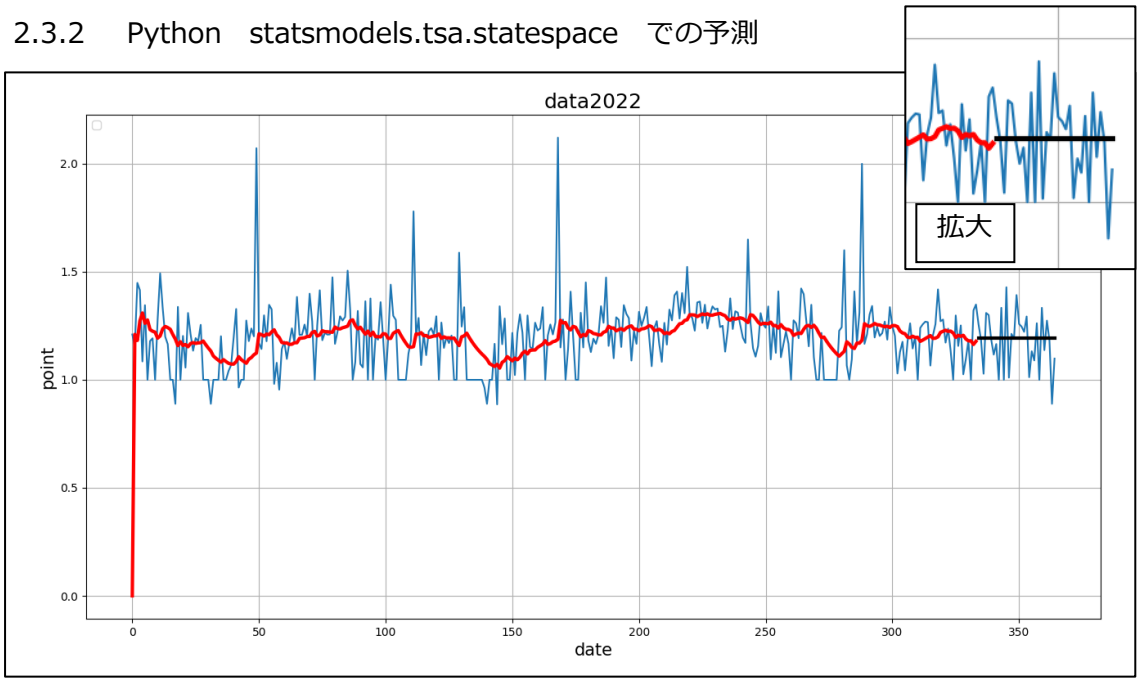
| | 対数尤度 | AIC |
|---|---|---|
| m=2 | -142.294 | -278.689 |
| m=15 | -147.128 | -262.255 |
| m=30 | -151.663 | -241.326 |

この表からは、m=2 が最適となりました。

（対数尤度は、大きい数値、AIC は小さい数値の方が、最適と考えます）

予測したグラフからは、335 から 337 までは予測できてますが、それ以降は、予測精度は、落ちてしますが、m が大きい程、振幅の減衰は少なく、より良く予測できてるように見えます。

### 2.3.2　Python　statsmodels.tsa.statespace　での予測




青線が観測値、赤線が推定値、黒線が予測値　です。

予測値は、観測値の振幅に追従することはなく、一定値で推移しています。

Local Level　モデルで、係数を 1 にしているので、予測値精度は落ちているようです。

2.3.3 Python　Pykalman, scipy.optimize,での予測

観測モデルを、トレンド成分（時間とともに単調に増加/減少する変動）と季節成分（同じ周期で規則的に繰り返される変動）と不規則成分との合計と考え、不規則成分は、AR モデルに準じると考えます。

数式で示すと

$y_n = t_n + s_n + r_n + w_n \qquad w_n \sim N(0, \delta^2)$

$t_n = \sum_{i=1}^{k} c_i^{(k)} t_{n-i} + \boldsymbol{v_{n1}} \qquad v_{n1} \sim N(0, \tau_1^2)$　トレンド成分

$s_n = -\sum_{i=1}^{p-1} s_{n-i} + \boldsymbol{v_{n2}} \qquad v_{n2} \sim N(0, \tau_2^2)$　季節成分

$r_n = \sum_{i=1}^{q} \Phi_i r_{n-1} + \boldsymbol{v_{n3}} \qquad v_{n3} \sim N(0, \tau_3^2)$　AR 成分

となり、状態モデル $x_n$ は　次のようなります。

$x_n = (t_n, t_{n-1}, \cdots t_{n-k+1}, s_n, s_{n-1}, \cdots s_{n-p+2}, r_n, \cdots r_{n-q+1})^T$

2-1 項にある状態空間モデルの式で、進めます。

$x_n = F_n x_{n-1} + G_n v_n$　(状態モデル)　$v_n \sim N(0, Q_n)$

$y_n = H_n x_n + w_n$　（観測モデル）　$w_n \sim N(0, R_n)$

ここで、トレンド成分の係数行列を $F_1$ , $G_1$, $H_1$ とし,季節成分の係数行列を $F_2$ , $G_2$, $H_2$ とし, AR 成分の係数行列を $F_3$ , $G_3$, $H_3$ とすると、

$$F_n = \begin{bmatrix} F_1 & 0 & 0 \\ 0 & F_2 & 0 \\ 0 & 0 & F_3 \end{bmatrix} \quad G_n = \begin{bmatrix} G_1 & 0 & 0 \\ 0 & G_2 & 0 \\ G_3 & 0 & 0 \end{bmatrix} \quad H^T = \begin{bmatrix} H_1^T \\ H_2^T \\ H_3^T \end{bmatrix}$$

で表現されるので、

データの特徴から　行列を定義します。（パラメータの推定は未だです）

トレンド成分は、傾きは無いとして、次数を 1 としました。係数は 1 です。

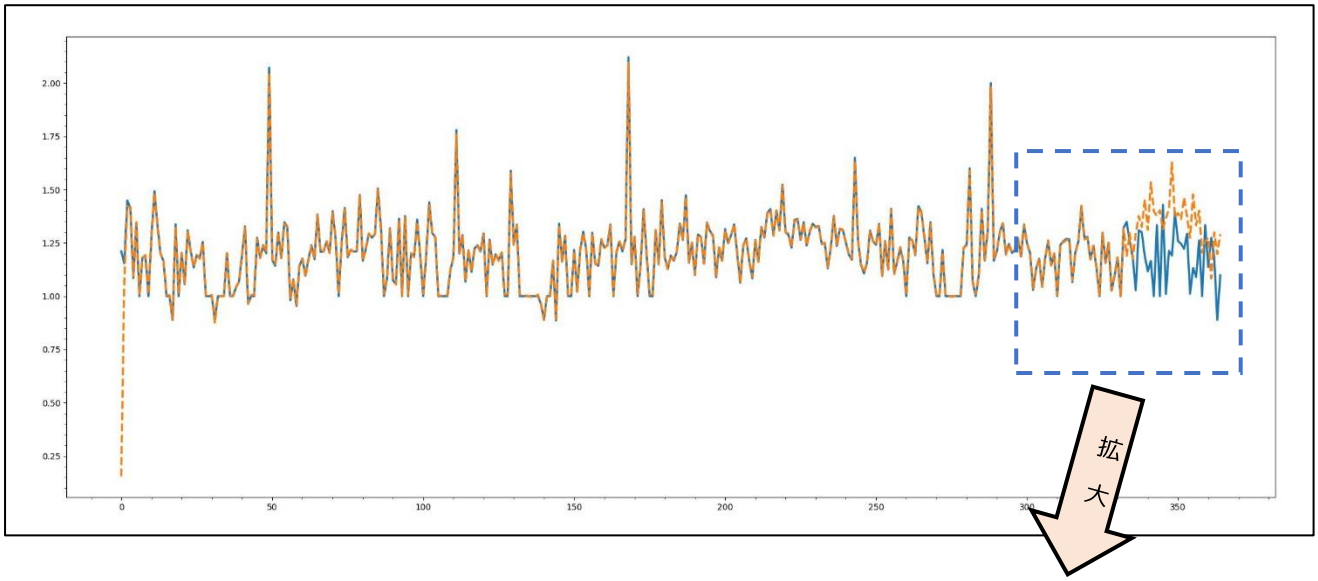
季節成分は、月単位の変化を想定して、次数を 30 としました。係数は 1 です。

AR 成分は、次数を 2 としました。係数は 0.5 です。

この条件で、Pykalman　で計算して、グラフを作成しました。

青線が、観測値で、橙色が推定値で、333 から 364　は予測値になります。

（4 月 1 日を 0 から起算しているので、1 日引いた数字になってます）

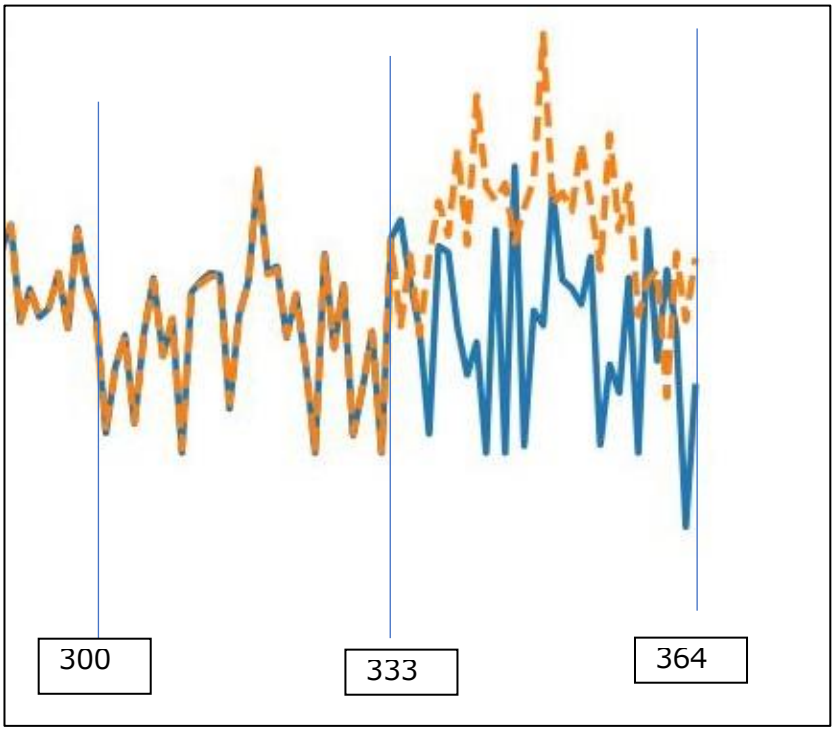



青線と橙線は、ほぼ重なり

訓練データへのあてはめは

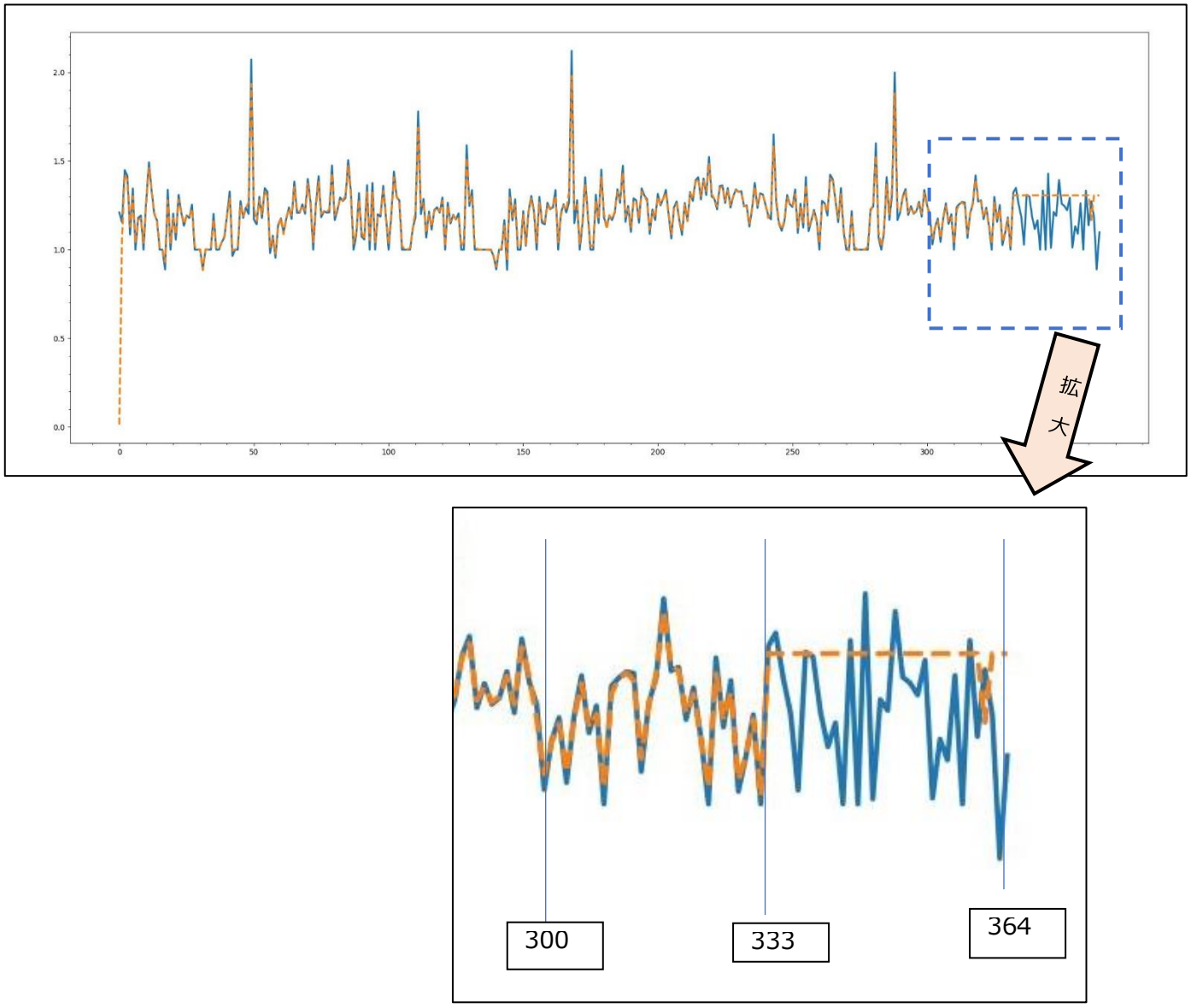
出来てますが、

予測は外れてしまいました。



さらに　scipy.optimize の minimize 関数を用いて、係数の推定を行い、

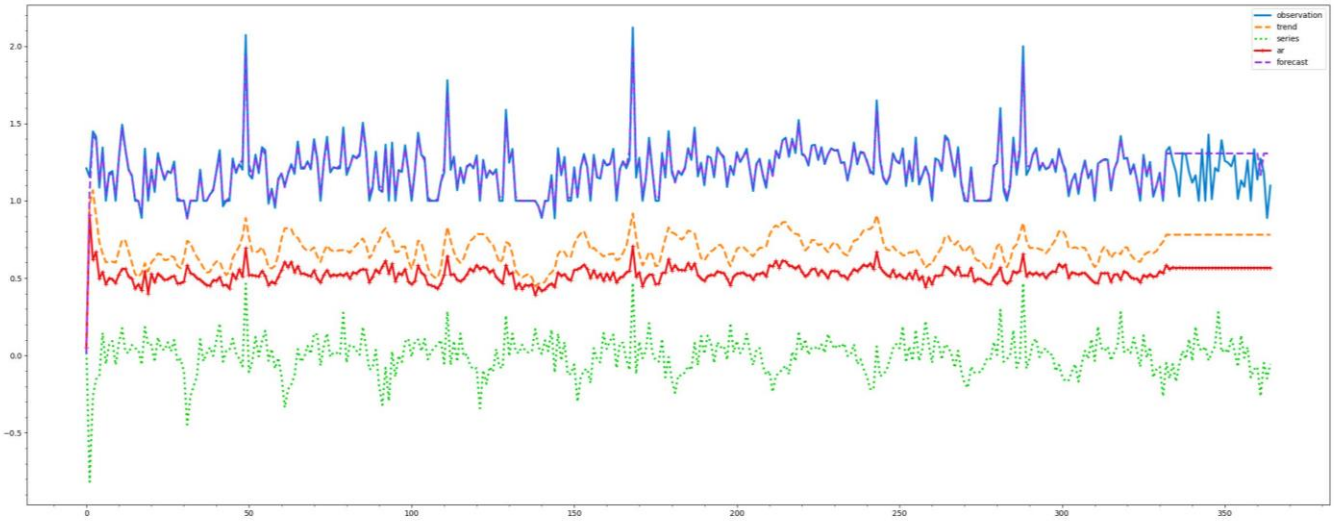
グラフを作成しました。




訓練データへのあてはめは出来てますが、予測は、分布の平均値であり、

振幅は、追従できてません。

グラフを成分別に分けたグラフが以下です。




下から、季節成分（緑線）、AR 成分（赤線）、トレンド成分（橙線）　となります。

3. まとめと今後の取組

　今回は、①R の TSSS、　②Python の statsmodels.tsa.statespace、

③Python の　Pykalman と scipy.optimize　による計算で、

いずれも、係数を推定し、カルマンフィルタにより推移を推定しますが、

いずれも、提供されるパッケージにある、ブラックボックスと言える関数による計算です。

　①と③は、訓練データのあてはめは出来たのですが、予測がうまくできませんでした。

②は、ローカルレベルモデルのためか、①、③　と比較して、精度は劣るようです。

　今後、モデルの定義、係数の見直し等に、取り組む予定です。