Pythonで体験

カルマン・フィルタ入門

ダウンロード・データあります

第6回 取得後のデータ処理なら…
より高精度な「カルマン・スムーサ」

ご購入はこちら

廣川 類

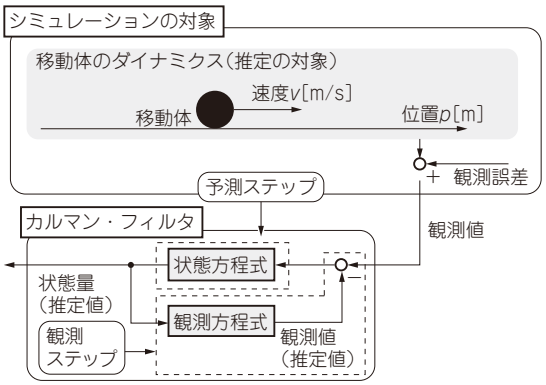

図1 1次元移動体シミュレーション・モデルとカルマン・フィルタ

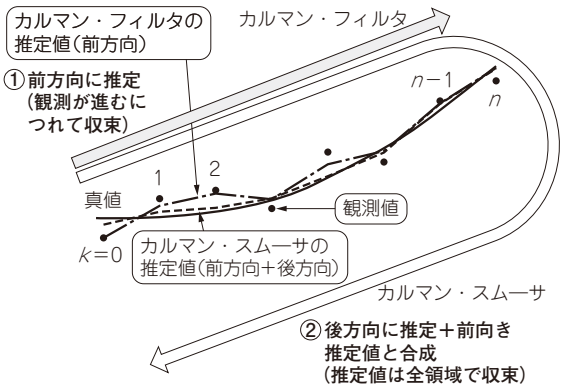

図2 カルマン・フィルタとカルマン・スムーサの処理方向の違い

カルマン・フィルタは，移動体のダイナミクスや動きの不確定性，センサ誤差の確率モデルを考慮して，最適な推定を行う強力なセンシング技術です．

今回は1次元移動体の運動(**図1**)を例に，カルマン・フィルタの応用として，カルマン・スムーサについて説明します．観測データを全て取得してから推定処理を行う場合には，カルマン・スムーサはカルマン・フィルタより良い精度が得られます．

フィルタとスムーサの違い

カルマン・フィルタが時間方向に前向きに逐次処理を行うのに対して，カルマン・スムーサは，カルマン・フィルタと同様に前向きに推定処理を行った後，再度，後ろ向きに推定処理を行って，推定結果を改善します(**図2**)．従って，逐次処理が可能なカルマン・フィルタは，リアルタイムの推定処理に適しています．一方，観測データを収集して，事後処理を行う用途に関しては，カルマン・フィルタを適用することも可能ですが，未来の観測エポックの伝搬も利用するカルマン・スムーサの方が，より良い精度が得られます．

今回は，カルマン・スムーサの処理アルゴリズムとして，最も利用されているRTS(Rauch, Tung and Striebel)スムーサを取り上げます．具体的には，定式化を行い，**図1**の1次元移動体の運動についてカルマン・フィルタとカルマン・スムーサを使った場合の推定結果を比べます．

RTSスムーサの定式化

RTSスムーサでは，通常のカルマン・フィルタの推定処理を行った後，最終ステップから時間軸を後ろ向きに処理していきます．次にそれぞれの処理内容を解説します．

● ①前向きの推定処理の数式(カルマン・フィルタ)

▶状態方程式

推定したいkステップ目の状態量ベクトル$\boldsymbol{x}_k$を次の状態方程式(離散系)で表します．

$$\boldsymbol{x}_k = \boldsymbol{\Phi}_k \boldsymbol{x}_{k-1} + \boldsymbol{w}_{q,\ k-1} \quad \cdots\cdots(1)$$

▶観測方程式

センサなどで観測されるkステップ目の観測量ベクトル$\boldsymbol{z}_k$を，状態量ベクトル$\boldsymbol{x}_k$から得るモデルを使って，次の観測方程式で表します．

$$\boldsymbol{z}_k = \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{w}_{n,\ k} \quad \cdots\cdots(2)$$

▶観測ステップと予測ステップの役割

カルマン・フィルタの計算は，状態方程式で次のス