Erklärung des Python Codes

Eingangsdaten

EightGroundTruth.csv --> Ground Truth

EightStepTimes.csv --> Schrittzeiten

nfg11.csv --> 5G Koordinaten hohe Frequenz hohe Genauigkeit

nfg53.csv --> 5G Koordinaten niedrige Frequenz niedrige Genauigkeit

nsh.csv --> Schrittlängen

nsl.csv --> Schrittrichtungen

Kalman Filter

MyKalmanFilterGNSSODO.py

main()

Ruft data_import() auf um die Daten einzulesen.

Es werden die Nullwerte festgelegt.

Ruft calc_delta() auf um Delta Werte zu berechnen aus Schrittlängen und Schrittrichtungen.

Ruft initial_covariance() auf um die Kovarianzmatrix der Parameter zu beginn zu definieren.

Ruft Kalman_Filter() auf den Kalman Filter durchlaufen zu lassen.

Ruft die Plot Funktion auf.

data_import()

Importiert Daten aus den .csv Dateien. Änderungen an den importierten Daten müssen hier vorgenommen werden.

Es werden die Daten zurückgegeben.

calc_delta()

Berechnet.

Es werden die Delta Werte zurückgegeben.

Kalman_Filter()

Läuft iterativ durch die Beobachtungsdaten (for Schleife).

Genauigkeit von Delta x und Delta y abhängig von Anzahl der Schritte (while Schleife).

Ruft prediction() auf.

Ruft correction() auf.

Es wird das Endergebnis zurückgegben.

prediction()

In dieser Funktion findet der Prediktionsschritt des Kalman Filters statt

Zurückgegeben wird der prädizierte x-Vektor und die dazugehörige Kovarianzmatrix.

correction()

In dieser Funktion wird die Innovation durchgeführt, die Kalman Gain Matrix erstellt und das Update berechnet.

Es wird der korrigierte x-Vektor zurückgegeben und die dazugehörige Kovarianzmatrix.

plot()

Funktion zum plotten der Ergebnisse

Least Square

ls5G.py

main()

Lädt Daten aus .csv Dateien ein.

Genauigkeitsparameter s1 und s2 werden definiert.

Funktion ls() wird aufgerufen

ls()

Läuft iterativ durch die Beobachtungen (for Schleife)

5G Beobachtungen werden für jeden Schritt simuliert durch addieren der Schrittlänge + Schrittrichtung auf die letzten 5G Koordinaten (while Schleife).

Ruft in jedem Schritt die WLS() Funktion auf.

Gibt die gesamte optimierte Trajektorie zurück.

WLS()

In dieser Funktion findet das berechnen der Trajektorie statt.

WLS gibt die optimierte Trajektorie für den einzelnen Schritt wieder aus.

plot()

Diese Funktion dient zum plotten der Ergebnisse

Extended Kalman Filter

MyExtendedKalmanFilterGNSSODO.py

main()

Ruft data_import() auf.

Definiert Startwerte.

Ruft calc_delta() auf.

Ruft initial_covariance() auf um die Kovarianzmatrix der Parameter zu beginn zu definieren.

Ruft Kalman_Filter().

data_import()

Importiert Daten aus den .csv Dateien. Änderungen an den importierten Daten müssen hier vorgenommen werden.

Es werden die Daten zurückgegeben.

calc_delta()

Berechnet.

Es werden die Delta Werte zurückgegeben.

Kalman_Filter():

Läuft iterativ durch die Beobachtungsdaten (for Schleife).

Schritte und Richtungen bis zur nächsten 5G Koordinate werden gesammelt (while Schleiife).

prediction() wird aufgerufen.

correction() wird aufgerufen.

Endergebnis wird zurückgegeben.

prediction()

Transitionsmatrix wird ensprechend der Anzahl der eingehenden Beobachtungen gebildet (for Schleife)

Zurückgegeben wird der prädizierte x-Vektor und die dazugehörige Kovarianzmatrix.

correction()

In dieser Funktion wird die Innovation durchgeführt, die Kalman Gain Matrix erstellt und das Update berechnet.

Funktion wird angepasst im Gegensatz zum diskreten Kalman Filter (siehe oben).

Die Kovarianzmatrix der Beobachtungen wird dynamisch, je nach Anzahl der Schritte zum nächsten 5G Punkt, angepasst (for Schleife).

Es wird das der korrigierte x-Vektor zurückgegeben und die dazugehörige Kovarianzmatrix.