Hochschule **Rosenheim**
University of Applied Sciences

Studiengang Produktionstechnik

DIPLOMARBEIT

Modellieren von idealen Messsignalen in Matlab/Simulink zur Verifikation von Navigationsalgorithmen zur Hagelbekämpfung

Vorgelegt von:	Maximilian Leinenbach
Matrikelnummer:	634939
am:	22.12.2011

zum
Erlangen des akademischen Grades

DIPLOMINGENIEUR

(Dipl.-Ing. (FH))

Erstprüfer: Prof. Dr.-Ing Peter Zentgraf, M.SC.

Zweitprüfer: Prof. Dr.-Ing. Franz Plötz

Selbständigkeitserklärung

Erklärung gemäß § 31/V RaPo

Hiermit erkläre ich, dass ich die von mir eingereichte Diplomarbeit zum Thema

Modellieren von idealen Messsignalen in Matlab/Simulink zur Verifikation von Navigationsalgorithmen zur Hagelbekämpfung

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Kolbermoor, den 22.12.2011

Unterschrift

Danksagung

Die vorliegende Diplomarbeit wurde im Labor für Mess- und Regelungstechnik der Hochschule Rosenheim durchgeführt. Mein besonderer Dank gilt Herrn Prof. Dr.-Ing Peter Zentgraf, M.SC. für die interessante Themastellung, die stete Förderung dieser Arbeit und die anregenden wissenschaftlichen Diskussionen.
Herrn Prof. Dr.-Ing. Franz Plötz danke ich für die Übernahme der Zweitkorrektur. Desweiteren möchte ich den Laboringenieuren Peter Viehauser und Martin Heigl für die Unterstützung während der gesamten Bearbeitungszeit danken. Mein letzter Dank gilt dem Bayer. Landesamt für Umwelt für die kostenfreie Bereitstellung der Messdaten.
In der Diplomarbeit werden die Vorgehensweisen und mathematischen Hintergründe der Berechnungen erklärt. Auf Darstellungen des kompletten Quelltextes wird weitgehend verzichtet, jedoch sind explizit Beispiele dargestellt, um die Arbeitsweise von Matlab zu verdeutlichen. Der Quelltext (M-Files) wird in digitaler Form auf der DVD vorhanden sein. Um entsprechende Funktionen des Quelltextes schneller zu finden, werden M-Files durch den in Matlab üblichen Namensschriftsatz gekennzeichnet. Beispielsweise würde die Funktion “test2view” im Text als `test2view` dargestellt werden.

Kurzfassung

Diplomarbeit im Studiengang Produktionstechnik

Maximilian Leinenbach

Modellieren von idealen Messsignalen in Matlab/Simulink zur Verifikation von Navigationsalgorithmen zur Hagelbekämpfung

Ziel dieser Diplomarbeit ist es den Bordrechner HAIL des Hagelflugzeugs mit künstlich generierten Messsignalen zu versorgen, um seine Algorithmen testen zu können, welche die Messdaten aufnehmen und verarbeiten. Dabei sollen Messsignale in physikalischen (SI) Einheiten generiert werden, welche ideale Sensoren liefern würden. Messsignale sollen alternativ auf zwei Arten generiert werden können, entweder durch Generieren der Messdaten mit Hilfe von physikalischen Gesetzen oder wahlweise durch Einlesen von realen, aufgezeichneten Flugdaten.
Die Implementierung dieses Softwaretools erfolgt auf Basis des Programms Matlab, welches für die Generierung der meteorologischen, flugmechanischen und geophysikalischen Messdaten bestens geeignet ist. Nach erfolgreicher Erstellung des Datensatzes können alle Messgrößen grafisch ausgewertet und wenn nötig, weiter bearbeitet werden. Zuletzt ist die Archivierung der Messflüge in eine MySQL Datenbank möglich.
Das Ergebnis ist eine Software, die eine vollautomatische Generierung von Messdatensätzen ermöglicht. Generierte Datensätze oder zukünftige Flugeinsätze können grafisch und adäquat bewertet werden. Auf Basis dieser Software ist der Aufbau einer Datenbank zur Grundlagenforschung an Großwetterlagen bei Hagelunwettern möglich.

Abstract

Diploma Thesis in Production Engineering
Maximilian Leinenbach
Modeling ideal measurement signals with Matlab/Simulink to verify navigation algorithms for controlling hail.

This diploma thesis aims to supply the hail-aircraft's on-board system HAIL with artificially generated measurement signals in order to be able to test its algorithms receiving and processing the measuring data. While doing so, measurement signals, which would provide ideal sensors, could be generated in physical units (SI). There will be two ways of generating these signals; either by using artificial measuring data based on physical laws, or by reading in authentic, recorded flight data. Implementation of this software tool is based on Matlab. The main task is the generation of meteorological, aeronautical and geophysical measuring data. After the successful creation of the (data) record, the results can be graphically analyzed and, if necessary, further processed. Measuring flights can be archived using an interface between Matlab and MySQL. This results in a software which allows a fully automatic generation of measuring data records. Thus, generated records or future flights can be graphically analyzed and appropriately assessed. This software could form the basis for creating a database for basic research of general weather situations during hail storms.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AgI	chemische Formel für Silberjodid
Char	Variable vom Typ Character (Zeichen)
CSV	Comma Separated Value, Datenformat für tabellarische Daten
DWD	Deutscher Wetterdienst
GK-HW	Gauß-Krüger - Hochwert
GK-RW	Gauß-Krüger - Rechtswert
GPS	Global Positioning System
Lat	Latitude - Geographische Breite
Lon	Longitude - Geographische Länge
String	Zeichenfolge von Character-Variablen

1 Einleitung

1.1 Hagelabwehr

Die Bildung von Hagel ist ein außergewöhnliches Wetterphänomen, welches bei dem Zusammentreffen von kalten und warmen Luftmassen geschehen kann. Durch starke Auf- und Abwinde an der Cumulonimbus Wolke werden Eis- und Schneekerne wiederholt zwischen den tieferen und höheren Wolkenschichten rasch hin- und herbewegt. Aufgrund der unterschiedlichen Temperaturbereiche lagern sich unterkühlte Wassertropfen solange an den Kristallisationskernen an, bis diese so schwer werden, dass der Aufwind nicht mehr genügend Kraft besitzt, die entstehenden Hagelkörner in der Wolke zu halten. Mit einer durchschnittlichen Fallgeschwindigkeit von über $25\frac{m}{s}$ bei einem Durchmesser von $20 - 40mm$, können die herabfallenden Hagelkörner einen enormen wirtschaftlichen Schaden anrichten, weshalb die Hagelabwehr eine sehr wichtige Rolle spielt. Ansatz der Hagelabwehr ist das Problem in der Entstehung zu beseitigen. Durch ein "Impfen" der Wolke mit Silberjodid-Kristallen, welche als künstliche Kristallisationskerne fungieren, soll die Anzahl der Hagelkörner erhöht werden. Durch die höhere Anzahl der Hagelkörner und der begrenzten Wassermenge in der Wolke entstehen kleinere Hagelkörner, die beim Flug zur Erdoberfläche auftauen und als Regen normalerweise keinen großen Schaden anrichten können. Die Wirksamkeit der Hagelbekämpfung ist jedoch noch nicht wissenschaftlich belegt. Beispielsweise kann bei einer Cumulonimbus Wolke auch trotz AgI-Impfung Hagel fallen. Der "Verein zur Erforschung der Wirksamkeit der Hagelbekämpfung im Raum Rosenheim e.V." hat es sich zur Aufgabe gemacht, dieses Wetterphänomen empirisch zu erforschen. Dies geschieht mit der systematischen Erfassung der Gewitterwolken bei Hagel-Wetterlagen, der umfangreichen exakten Datensammlung über Hageleinsätze und ihren Auswirkungen, sowie der Auswertung der gesammelten Daten, etwa über Vergleiche mit Gebieten ohne Hagelabwehr.

1.2 Ausgangssituation

Das Projekt RO-BERTA, mit welchem die Hochschule Rosenheim von dem Hagelforschungsverein beauftragt wurde hat zur Aufgabe, mit Hilfe der Radardaten des DWD, einen dreidimensionalen Blick durch eine Hagelwolke zu ermöglichen und dies dem Piloten ins Cockpit zu projizieren, die visuelle Darstellung von Messdaten, sowie den Aufbau einer Datenbank zur Grundlagenforschung an Großwetterlagen bei Hagelunwetter.
In der Diplomarbeit von Tobias Höglauer[1] wurde bereits ein Programm zur Visualisierung von

[1] Aufbereitung und Online- Visualisierung der Messdaten eines Hagelabwehrflugs, 2009

Kurs, Temperatur, Druck, Luftfeuchte, DWD Radardaten und AgI-Intensität der Kanonen via Google Earth geschaffen. Mit dem zukünftigen Bordrechner HAIL (Hagel Abwehr in der Luft) wird es möglich sein, eine höhere Anzahl von Messgrößen aufnehmen zu können. Ausserdem ist es zweckmässig ein anderes Tool zu erstellen, das besser in das Gesamtkonzept von RO-BERTA passt.

1.3 Zielsetzung

Primäres Ziel dieser Diplomarbeit ist es den Bordrechner HAIL des Hagelflugzeugs mit künstlich generierten Messsignalen zu versorgen, um seine Algorithmen testen zu können, welche die Messdaten aufnehmen und verarbeiten. Dabei sollen Messsignale in physikalischen (SI) Einheiten generiert werden, welche ideale Sensoren liefern würden. Messsignale sollen alternativ auf zwei Arten generiert werden können, entweder durch Generieren der Messdaten mit Hilfe von physikalischen Gesetzen oder wahlweise durch Einlesen von realen, aufgezeichneten Flugdaten. Die Messdaten gliedern sich wie folgt:

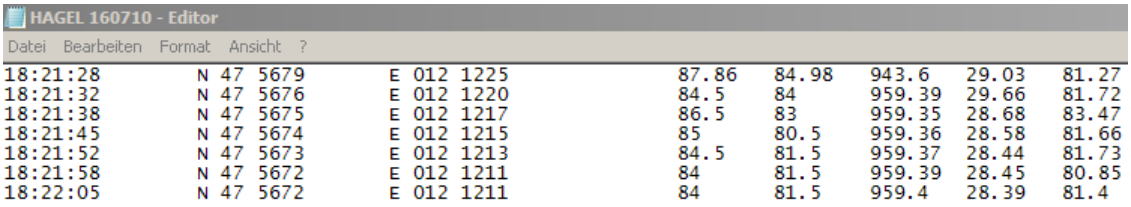
- **flugmechanische Messgrößen:** Geschwindigkeit und Beschleunigung, Winkelgeschwindigkeit und Beschleunigung, virtuelle Beschleunigungssensoren, Position und Lage des Flugzeugs in Bezug auf die Erdoberfläche
- **meteorologische Messgrößen:** Temperatur, Luftdruck (dynamisch und statisch), Luftfeuchte, Windstärke und -Richtung
- **geophysikalische Messgrößen:** elektrische Feldstärke, magnetische Feldstärke
- **sonstige Messgrößen:** Feinstaubdichte, Hagelwolke

Ein weiterer wichtiger Aspekt ist die grafische Auswertung der Messdaten. Mithilfe des Programms sollen nicht nur die künstlich generierten Datensätze ausgewertet werden, auch für ältere Flugaufzeichnungen und zukünftige Flüge muss eine Auswertung möglich sein. Als letzte Aufgabe ist eine Schnittstelle zwischen Matlab und MySQL Datenbank zu erstellen. Hierbei geht es primär darum aufzuzeigen, welche Möglichkeiten bestehen und ob die Datenübertragung der Anforderung des Projektes RO-BERTA gewachsen ist. Die Bedienbarkeit des Softwaretools soll nicht aufwendig und auch für ungeübte Benutzer geeignet sein. Die Veränderung bzw. Erweiterung einzelner Programmteile muss ebenso gewährleistet werden. Die Umsetzung erfolgt auf der Basis des Programmes Matlab.

2 Aufbereitung der Messdaten

2.1 Allgemein

Als Berechnungsgrundlage aller Messdaten dienen Aufzeichnungen des früheren Bordrechners Ro-Bert. Die Datenstruktur ist wie folgt aufgebaut: In der ersten Spalte wird der Zeitpunkt der Messdatenaufzeichnung festgehalten. In den darauffolgenden beiden Spalten sind Angaben zum Kurs, welche die geographische Breite und Länge beschreiben. Spalte vier ist ein Platzhalter, fünf und sechs beschreiben die Intensität der Silberjodidkanonen und die letzten drei Spalten beschreiben Druck, Temperatur und Luftfeuchte. Der Dateiname enthält Angaben zum Aufzeichnungsdatum.


HAGEL 160710 - Editor

Datei Bearbeiten Format Ansicht ?

18:21:28	N 47 5679	E 012 1225	87.86	84.98	943.6	29.03	81.27
18:21:32	N 47 5676	E 012 1220	84.5	84	959.39	29.66	81.72
18:21:38	N 47 5675	E 012 1217	86.5	83	959.35	28.68	83.47
18:21:45	N 47 5674	E 012 1215	85	80.5	959.36	28.58	81.66
18:21:52	N 47 5673	E 012 1213	84.5	81.5	959.37	28.44	81.73
18:21:58	N 47 5672	E 012 1211	84	81.5	959.39	28.45	80.85
18:22:05	N 47 5672	E 012 1211	84	81.5	959.4	28.39	81.4

Abb. 2.1: Ausschnitt Ro-Bert Datensatz

Der Programmablauf A, welcher bestehende Datensätze auswertet, verwendet alle Messdaten und berechnet daraus noch fehlende Größen wie z.B. Flughöhe.
In den Programmabläufen B und C, welche das Generieren komplett neuer Datensätze ermöglichen, werden nur Angaben zu Flugbahn und Zeit weiterverwendet. Es bestünde zwar die Möglichkeit selbst Flugbahnen zu definieren, dies ist aber extrem aufwendig und steht nicht im Verhältnis zum Nutzen. Als Basis stehen neun vorbereitete Flugbahnen zur Verfügung.

2.2 Matlab

Abb. 2.2: Ablauf - Datenaufbereitung

2.2.1 Importieren

Für weitere Berechnungen muss zunächst ein Datensatz in Matlab importiert werden. Die Funktion `Robert_Importskript` bietet die Möglichkeit, den Importierungsvorgang automatisch auszuführen. Dabei wird die Information des Datums in einer Variablen des Typs `string` im Workspace hinterlegt. Zeitstempel und Kurs werden in einer Variablen des Typs `cell` und die restlichen Messdaten in einer Matrix des Datenformates `double` im Workspace gespeichert.

2.2.2 Konvertieren

Mathematische Berechnungen werden in Matlab nur mit numerischen Datentypen durchgeführt. Anfangs ist es nötig, Zeitstempel und Kurs in eben dieses Format zu konvertieren, da sie gebündelt in einer Cellvariablen im String-Format vorliegen.

2.2.2.1 Zeit

Um mit Zeitwerten rechnen zu können gibt es die Möglichkeit, Zeitwerte in ein eigenes numerisches Matlabzeitformat zu konvertieren. Mittels den Befehlen `datenum` und `datestr` können Zeitwerte konvertiert werden. Beide Funktionen können vektoriell arbeiten, es wird gleich der komplette Vektor umgewandelt.
Beispiel:

```
%
%Beispiel Matlabzeit
%
%Timestr     = Zeitangabe [String]
%Time        = Zeitangabe [Matlab: numerischer Zeitwert]
Timestr='13:00:00';
Time=datenum(Timestr,'HH:MM:SS');
Timestr2=datestr(Time,'HH:MM:SS')
Timestr2
    '13:00:00'
```

2.2.2.2 Kurs

Für die Umwandlung der geographischen Längen- und Breitengrade ist zunächst die Anpassung der Strings notwendig, da die alphanumerische Darstellung keine direkte Umwandlung in einen Zahlenwert erlaubt. Mit der Funktion `corstr2num` werden die entsprechenden Werte zunächst in einen anderen String übermittelt und angepasst. Anschließend wird der neu entstandene String in das Datenformat double konvertiert. Für die weiteren Berechnungen werden die geographischen Längen- und Breitengrade noch in Dezimalgraddarstellung gebracht.

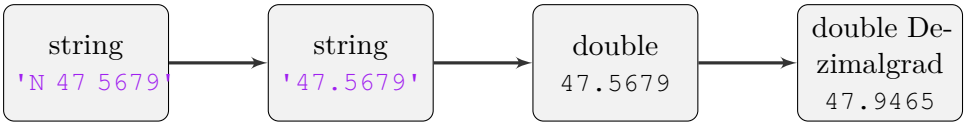


Abb. 2.3: Längen- und Breitengrad konvertieren

2.2.3 Aufbereitung

2.2.3.1 Zeit

Die Samplerate von HAIL wird höher sein, als die des alten Bordrechners mit ca. 8 Sekunden. Aufgrund dessen wurde beschlossen, dass die Samplerate der generierten Daten bzw. ausgewerteten Daten 0.1 Sekunden betragen muss. Erster Schritt ist nun, den importierten Zeitvektor in die neue Samplerate umzuwandeln. Mit der Funktion `Timestep1s` wird außerdem noch ein zweiter Zeitvektor erstellt, der von 00:00:00:000 bis zum Ende der Messung ebenfalls im 0.1 Sekundentakt hochzählt.

2.2.3.2 Messdaten

Durch lineare Interpolation werden Messwerte erzeugt, die eine Auflösung von 0.1 Sekunden zwischen einer Messerfassung haben. Die Interpolation aller Messwerte eines Datensatzes geschieht mit der Matlab Funktion `interp1`. Zur Verdeutlichung wird der Vorgang für die Größe Temperatur demonstriert.

```
%
%Beispiel Temperaturinterpolation
%
%Time               = Zeitvektor Samplerate Ro-Bert
%Timesteps          = Zeitvektor Samplerate 0.1s
%data(1:end,5)      = Temperaturvektor aus der Matrix
%                     des importierten Ro-Bert Datensatzes
%Temperatur.T_Flug  = Temperaturvektor mit Messdaten in 0.1s
%                     Takt
Temperatur.T_Flug= interp1(Time,data(1:end,5),Timesteps);
```

Als Ergebnis erhält man einen Vektor, dessen Länge gleich der Länge des Zeitvektors mit Sampelrate 0.1 Sekunde ist.

Ro-Bert		**Interpoliert**	
Zeit:	Temperatur:	Zeit:	Temperatur:
18:21:28	29.03	18:21:28:000	29,0300000000000
18:21:32	29.66	18:21:28:100	29,0457499207910
18:21:38	28.68	18:21:28:200	29,0614998415820
18:21:45	28.58	18:21:28:300	29,0772497623730
18:21:52	28.44	18:21:28:400	29,0929996831639
18:21:58	28.45	18:21:28:500	29,1087496039549
⋮	⋮	⋮	⋮

Tab. 2.1: Interpolation der Messdaten

2.2.3.3 Gauß-Krüger - Koordinatensystem

Da sich die Dezimalgradstellung nicht für Berechnungen eignet, werden die Längen- und Breitengrade in das Gauß-Krüger-Koordinatensystem überführt. Bei dem Gauß-Krüger-Koordinatensystem handelt es sich um ein kartesisches Koordinatensystem, das es ermöglicht, kleine Gebiete der Erde mit metrischen Koordinaten (Rechtswert und Hochwert), deren Einheit Meter ist, darzustellen. Möglich wird diese Kartenprojektion durch das von Gauß entwickelte Verfahren der winkeltreuen Abbildung mit längentreuem Meridian. Hierbei wird ein Zylinder senkrecht zur Erdachse über die Erde gezogen, wobei der Zylinderumfang gleich dem Erdumfang ist. Projeziert man den Erdoberflächenbereich östlich und westlich des Bezugsmeridian auf den Zylinder, nimmt jedoch der Fehler mit zunehmenden Abstand zum Bezugsmeridian zu. Bei der Großkreisprojektion, welche auch das Gauß-Krüger-Koordinatensystem benüzt, ist die Fehlertoleranz nach 3° Abweichung zum Bezugsmeridian erreicht.

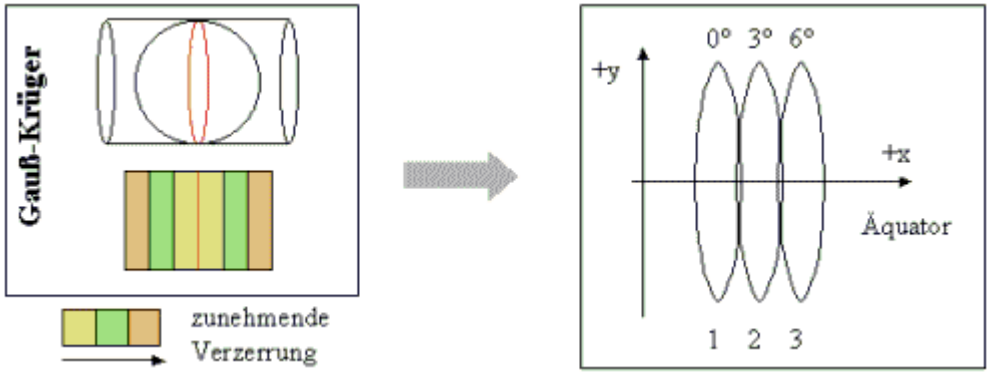


Abb. 2.4: Gauß-Krüger - Großkreisprojektion
Bildquelle: http://rzv037.rz.tu-bs.de/gis/gis/drucken/ft4.htm

Durch Neupositionieren des Zylinders auf den nächsten Bezugsmeridian umgeht man eine zu hohe Fehlertoleranz. Beispielsweise besitzt die Fläche von Deutschland drei Bezugsmeridiane. Die Funktionen[1] `geo2gk` und `gk2geo` ermöglichen das Konvertieren in beide Richtungen.
Das Bezugssystem - Erdsystem wird in der Diplomarbeit folgendermaßen definiert. Auf der Abszisse werden die Rechtswerte in Richtung geografischer Osten, auf der Ordinate die Hochwerte in Richtung geografischer Norden und auf der Applikate die Höhe über Normalnull in Richtung Himmel aufgetragen. Alle vektoriellen Größen werden sich, falls nicht anders angegeben, auf dieses Koordinatensystem beziehen.

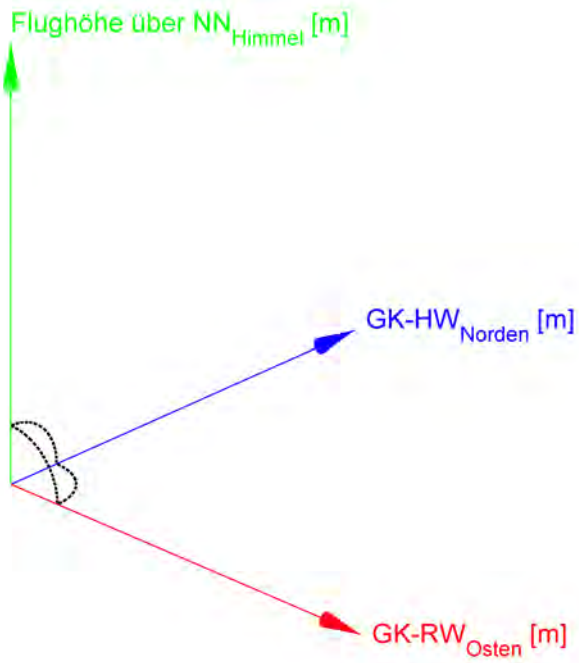


Abb. 2.5: Erdkoordinatensystem

[1] Prof. Dr.-Ing Peter Zentgraf, M.SC., 02.08.2010

3 Generierung der Messsignale

3.1 Allgemein

Die Grundlage bei der Generierung der Messsignale ist eine der neun vorbereiteten Flugbahnen. Die Flugbahn enthält vier Zeilenvektoren mit Angaben zur Position im kartesischen Gauß-Krüger-Koordinatensystem, sowie der Flugzeit für jeden Ort.

Flugzeit	GK-Rechtswert [m]	GK-Hochwert [m]	Flughöhe [m]
18:21:28	4515346	5312057	468
18:21:36	4515346	5312057	468.30
⋮	⋮	⋮	⋮

Ein Teil der generierten Messsignale ist nicht nur eine ortsabhängige sondern auch eine zeitabhängige Größe. Beispielsweise sinkt die Temperatur der Luft nicht nur mit der Höhe, sondern auch Nachts. Um eine möglichst genaue Simulation dieses zeit- und ortsabhängigen Verhaltens zu schaffen, muss zunächst ein Startzeitpunkt definiert werden. Der Startzeitpunkt enthält Angaben über Datum und Zeit.
Der bestehende Flugzeitvektor muss mit der neuen Startzeit verknüpft werden. Durch die Zeitabhängigkeit der Flugzeugposition ist es jedoch wichtig, dass $\Delta Zeit$ zwischen den Vektorelementen der Flugzeit nicht verändert wird. Eine Veränderung würde zu unrealistischen Ergebnissen bei Geschwindkeit und den daraus abgeleiteten Größen führen. Mit Hilfe der Funktion `Zeiteinteilung` kann dem Vektor der Flugzeit problemlos eine neue Startzeit wie z.B. 15:00:00 Uhr zugeteilt werden.

Flugzeit	Flugzeit Neu	$\Delta Zeit$ [s]
18:21:28	15:00:00	8
18:21:36	15:00:08	7
18:21:43	15:00:15	⋮
⋮	⋮	⋮

Als letzten Schritt vor der eigentlichen Generierung der Messdaten muss noch die Bodenspur der Flugbahn berechnet werden. Die Bodenspur gibt die Geländehöhe über Normalnull an. Da ein digitales Geländemodell[1], welches die Erdoberfläche als eine in der Lage und Höhe

[1] Geobasisdaten © Bayerische Vermessungsverwaltung 2010

bekannte Punktwolke beschreibt, vorhanden ist, kann die Bodenspur interpoliert werden. Das Geländemodell hat eine Rasterung von 25 Metern und bezieht sich ebenfalls auf das Gauß-Krüger-Koordinatensystem. Die Punktewolke muss zunächst jedoch in das gebräuchliche Inputformat des Befehls `surf` gebracht werden, das heißt, die Angabe zu GK-Rechtswert und GK-Hochwert muss in einem jeweils monoton steigenden Zeilenvektor gespeichert werden. Die Angabe zur Geländehöhe wird in einer Matrix hinterlegt. Die Größe der entstehenden Matrix ist durch die Länge der Vektoren definiert. Je nach Größe des Ausschnittsbereiches des digitalen Geländemodelles ergibt sich somit für GK-Rechtswert(nx1), GK-Hochwert(mx1) und Geländehöhe(mxn).
Mithilfe der Funktion `points2mat`[2] werden die Koordinatenpunkte der Wolke geordnet und in das entsprechende Format konvertiert.

```
%
%Beispiel points2mat
%
%ungeordnete Punkte
p1=[1;1;0];
p2=[2;1;1];
p3=[1;2;3];
p4=[2;2;2];
p5=[3;1;1];
p6=[3;2;3];
%x,y und z Komponente der Punkte
for n=1:6
    xp(n,1)=eval(sprintf('p%d(1,1);', n));
    yp(n,1)=eval(sprintf('p%d(2,1);', n));
    zp(n,1)=eval(sprintf('p%d(3,1);', n));
end
%Surf Darstellung über points2mat
[xVec,yVec,zMat]=points2mat(xp,yp,zp)
xVec
    1
    2
    3
yVec
    1
    2
zMat
    0 1 1
    3 2 3
```

[2] Prof. Dr.-Ing Peter Zentgraf, M.SC., 28.07.2010

Das Ergebnis ist so zu interpretieren, dass durch die beiden Vektoren alle x und y Komibinationen der Punkte möglich sind. Durch die Position eines Elementes in den Vektoren kann aus der Matrix der zugehörige z-Wert ermittelt werden, z.B. Punkt p4.

$$\begin{pmatrix} xVec(n,1) \\ yVec(m,1) \\ zMat(m,n) \end{pmatrix} n \to 2, m \to 2 = \begin{pmatrix} xVec(2,1) \\ yVec(2,1) \\ zMat(2,2) \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix} = p4 \tag{3.1}$$

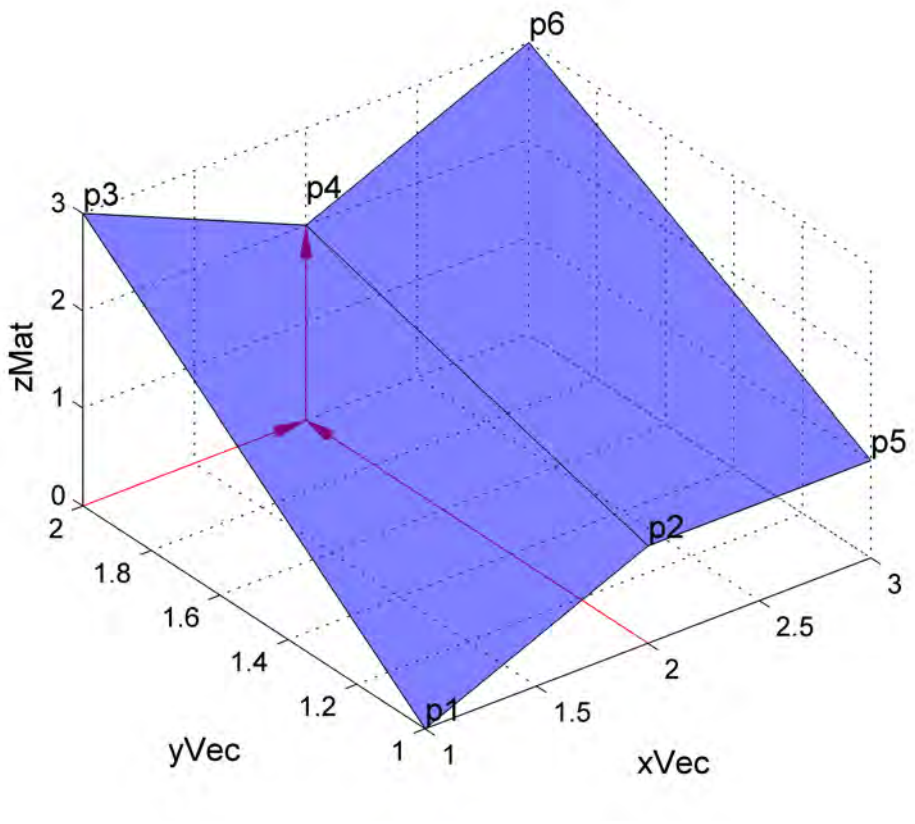


Abb. 3.1: Darstellung points2mat

Die Berechnung der Bodenspur wird mittels `interp2` durchgeführt. Zu dem gegebenen Flugkurs wird nun die unbekannte Höhenkomponente aus dem digitalen Geländemodell interpoliert. Die Syntax des Befehles lautet `zi=interp2(xVec,yVec,zMat,xi,yi);`, wobei die Variablen xVec, yVec und zMat das Geländemodell darstellen. Die Variablen xi und yi repräsentieren die Flugbahn in der xy-Ebene. Der Ablauf ist in Abbildung 3.2 noch einmal dargestellt.

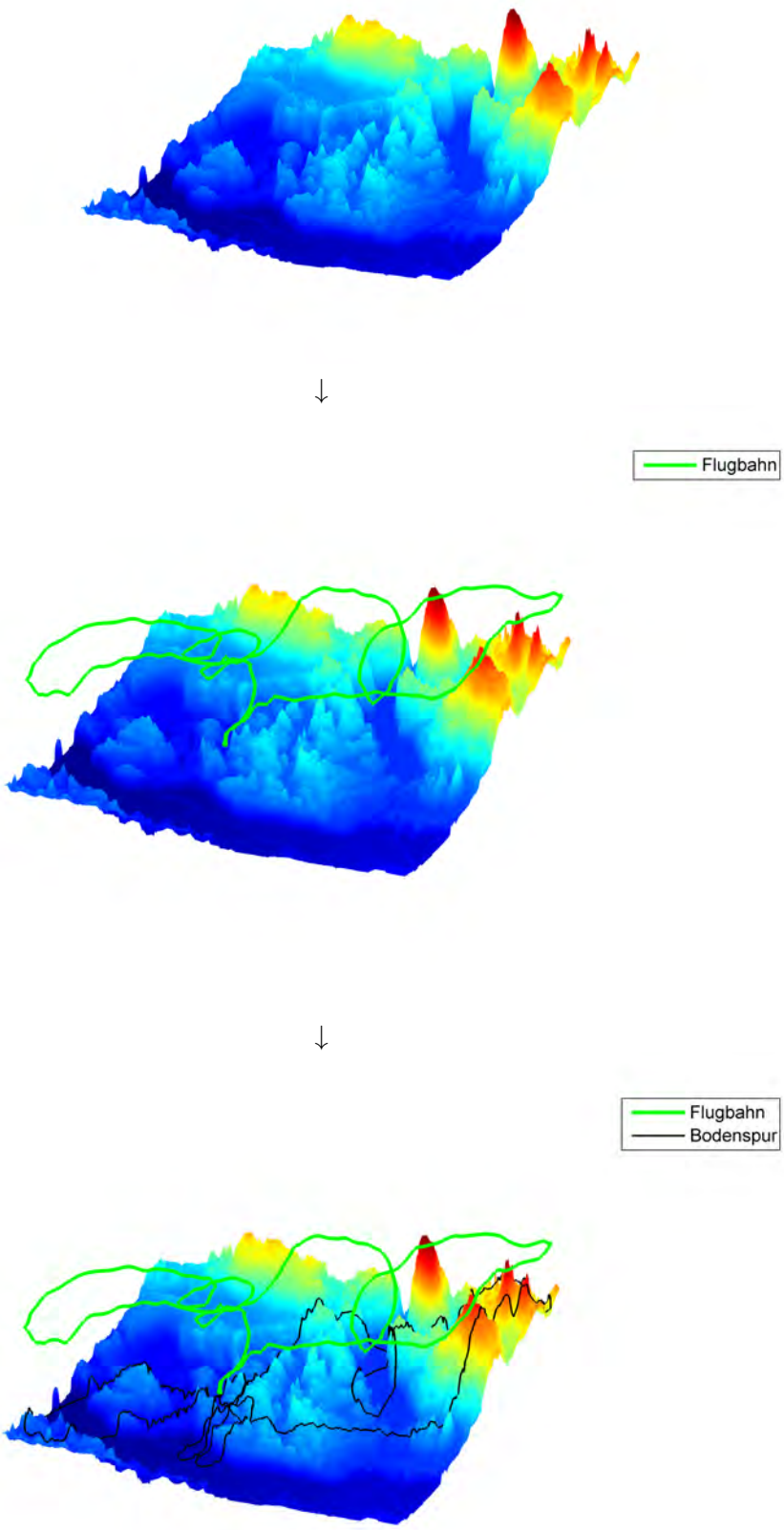


Abb. 3.2: Interpolation der Bodenspur

3.2 flugmechanische Signale

Die flugmechanischen Signale beschreiben die Kinematik und Lage des Flugzeugs im Bezugssystem/Erdsystem. Zu den Signalen zählen: Geschwindigkeit, Beschleunigung, Winkelgeschwindigkeit, Winkelbeschleunigung sowie die Drehmatrix im R^3. Alle kinematischen Größen werden anhand numerischer Verfahren berechnet, durch die Samplerate von 0.1 Sekunden reicht die Genauigkeit jedoch aus.
Bei der Berechnung wird darauf geachtet, dass die Algorithmen ressourcensparend programmiert werden, da mit großen Datenmengen gearbeitet wird. Matlab bietet eine Vielzahl von bereits definierten Befehlen, die den Einsatz von Schleifen überflüssig machen. Beispielsweise kann die Differenz benachbarter Werte mit `diff` anstelle einer `for−Schleife` geschehen.

```
%Berechnung Differenz Vektor Werte
x=[1;2;3;4]
%Matlabfunktion
x_diff_m=diff(x);
%Schleife
i=length(x);
for n=1:i-1
    x_diff_s(n,1)=x(n+1,1)-x(n,1);
end
%Prüfen ob Ergebnisse identisch sind.
x_diff_m==x_diff_s
ans=
    1
    1
    1
```

Um die Länge der Gleichungen nicht unnötig zu vergrößern, gilt:

$$\begin{pmatrix} Rechtswert \\ Hochwert \\ Flughöhe \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3.2}$$

3.2.1 Geschwindigkeit

Die Geschwindigkeit gibt die Ortsveränderung in einem Zeitabschnitt an. Sie wird über die Ortsvektoren $r(t)$ der Flugbahnpunkte berechnet. Es wird angenommen, dass sich der Flugzeugneutral direkt auf der Flugbahn bewegt. In Matlab kann die Berechnung mit der Funktion `velocity` durchgeführt werden.

$$v(t) = \frac{r(t_{n+1}) - r(t_n)}{t(t_{n+1}) - t(t_n)} = \frac{\Delta r}{\Delta t} \left[\frac{m}{s}\right] \tag{3.3}$$

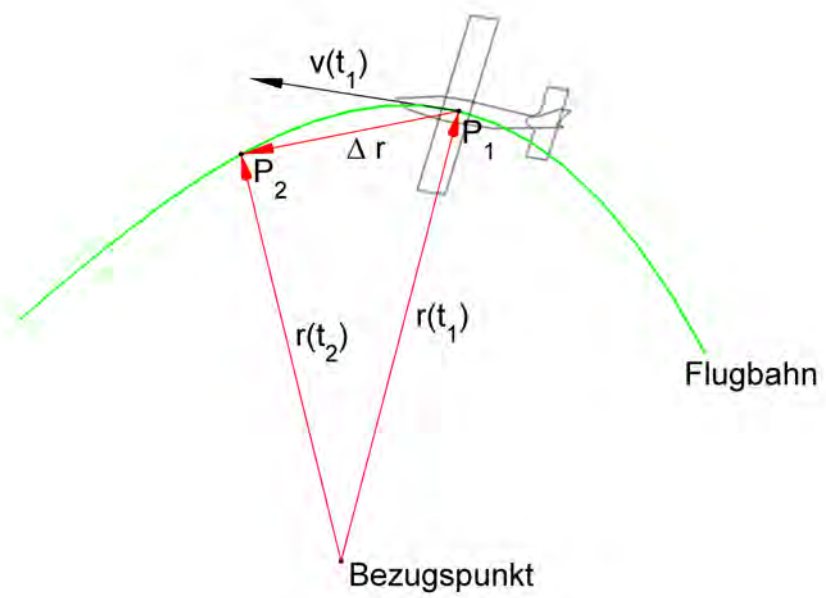


Abb. 3.3: Berechnung Geschwindigkeit

Für die Auswertung ist nicht nur die vektorielle Größe wichtig, sondern auch der Betrag der Geschwindigkeit. Bei einer Samplerate von 0.1 Sekunden erfolgt die Berechnung der Geschwindigkeit gemäß der Formeln 3.4 und 3.5.

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \frac{\begin{pmatrix} x(t_{n+1}) \\ y(t_{n+1}) \\ z(t_{n+1}) \end{pmatrix} - \begin{pmatrix} x(t_n) \\ y(t_n) \\ z(t_n) \end{pmatrix}}{0.1} \left[\frac{m}{s^2}\right] \tag{3.4}$$

$$|v| = \sqrt{v_x^2 + v_y^2 + v_z^2} \left[\frac{m}{s}\right] \tag{3.5}$$

3.2.2 Beschleunigung

Die Beschleunigung gibt die Geschwindigkeitsveränderung in einem Zeitabschnitt an. Sie wird über die Geschwindigkeitsvektoren $v(t)$ des Flugzeugs berechnet. In Matlab kann die Berechnung mit der Funktion `acceleration` durchgeführt werden.

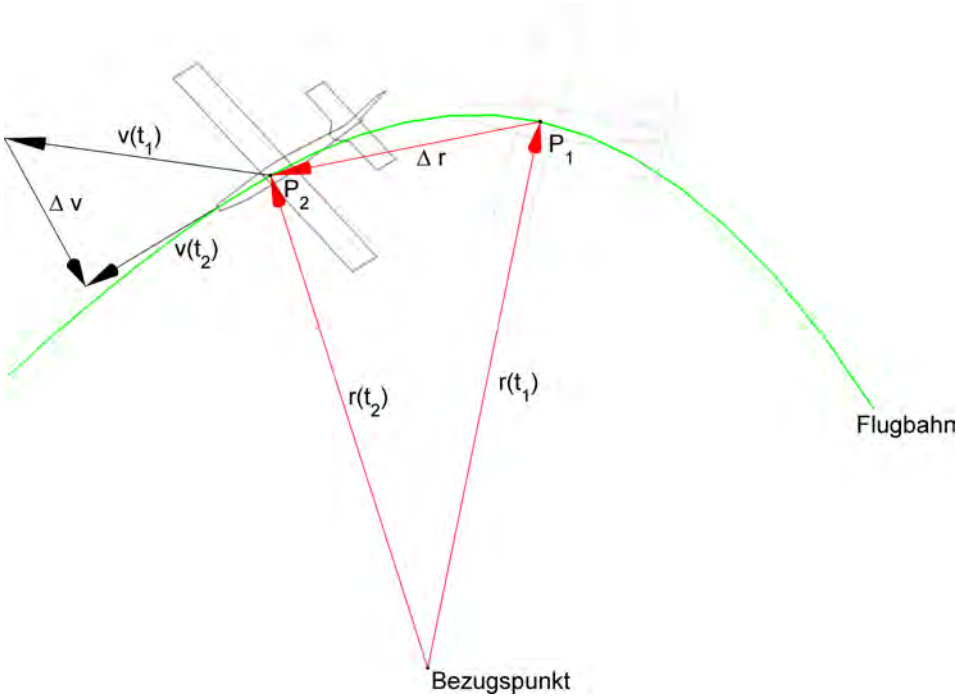


Abb. 3.4: Berechnung Beschleunigung

$$a(t) = \frac{v(t_{n+1}) - v(t_n)}{t_{n+1} - t_n} = \frac{\Delta v}{\Delta t} \left[\frac{m}{s^2}\right] \tag{3.6}$$

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \frac{\begin{pmatrix} v_x(t_{n+1}) \\ v_y(t_{n+1}) \\ v_z(t_{n+1}) \end{pmatrix} - \begin{pmatrix} v_x(t_n) \\ v_y(t_n) \\ v_z(t_n) \end{pmatrix}}{0.1} \left[\frac{m}{s^2}\right] \tag{3.7}$$

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2} \left[\frac{m}{s^2}\right] \tag{3.8}$$

3.2.3 Lage des Flugzeugs

3.2.3.1 Allgemein

Die Lage des Flugzeugs beschreibt die Verdrehung des flugzeugeigenen Koordinatensystems zum erdfesten Koordinatensystem. Die Verdrehung des Flugzeugkoordinatensystems kann mit einer Drehung R um die x-Achse (Rollen), y-Achse (Nicken) und z-Achse (Gieren) beschrieben werden.

$$R_x(\Phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\Phi) & -sin(\Phi) \\ 0 & sin(\Phi) & cos(\Phi) \end{pmatrix} \tag{3.9}$$

$$R_y(\Theta) = \begin{pmatrix} cos(\Theta) & 0 & sin(\Theta) \\ 0 & 1 & 0 \\ -sin(\Theta) & 0 & cos(\Theta) \end{pmatrix} \tag{3.10}$$

$$R_z(\Psi) = \begin{pmatrix} cos(\Psi) & -sin(\Theta) & 0 \\ sin(\Psi) & cos(\Theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

$$R = R_z(\Psi) \cdot R_y(\Theta) \cdot R_x(\Phi) \tag{3.12}$$

Die Transformationsmatrix beschreibt die Drehung des Flugzeugsystems um den Koordinatenursprung im normierten Vektorraum .

$$R = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{pmatrix} \tag{3.13}$$

Die Ausrichtung der Achsen des gedrehten Koordinatensystems lassen sich durch die drei Vektoren x_F, y_F und z_F darstellen.

$$x_F = \begin{pmatrix} r_{1,1} \\ r_{2,1} \\ r_{3,1} \end{pmatrix}, y_F = \begin{pmatrix} r_{1,2} \\ r_{2,2} \\ r_{3,2} \end{pmatrix}, z_F = \begin{pmatrix} r_{1,3} \\ r_{2,3} \\ r_{3,3} \end{pmatrix} \tag{3.14}$$

In Abbildung 3.5 wird der Zusammenhang visualisiert. Es gilt $_E$= Erdkoordinatensystem und $_F$= Flugzeugkoordinatensystem.

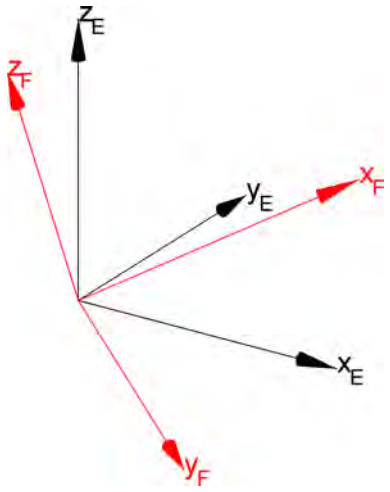


Abb. 3.5: Erdsystem und Flugzeugsystem

3.2.3.2 Algorithmus

Ziel des Algorithmus ist es, die drei Richtungsvektoren x_F, y_F und z_F anhand gegebener Größen zu berechnen. Damit ein Algorithmus definiert werden kann, müssen zunächst Annahmen getroffen werden.

- Die Ausrichtung der x-Achse ist tangential zur Flugbahn.
- Die Drehung um die x-Achse (Rollen) wird aus dem Verhältnis der Beschleunigung und der Erdanziehung gebildet. Die Zentrifugalkraft wird durch das Rollen kompensiert.
- Das Flugzeug führt keine akrobatischen Manöver wie Looping oder Schraube aus.

Für die Bestimmung der Transformationsmatrix gilt desweiteren, dass sich alle Vektoren im normierten Vektorraum befinden. Die Länge der Vektoren beträgt 1. Für die Berechnung ist die geometrische Bedeutung des Kreuzproduktes zweier Vektoren entsprechend wichtig. Das Kreuzprodukt der beiden Vektoren ist ein dritter Vektor, der senkrecht auf der von den beiden Vektoren aufgespannten Ebene steht und mit ihnen ein Rechtssystem bildet. Dies bedeutet, dass nur zwei Achsen vollständig definiert sein müssen um ein Koordinatensystem zu erzeugen, weil die dritte Achse per Kreuzprodukt berechnet wird.
Die tangentiale Ausrichtung der x-Achse zur Flugbahn kann durch den Geschwindigkeitsvektor beschrieben werden (Abbildung 3.3). In Matlab kann die Berechnung mit der Funktion `DrehmatrixEulerwinkel` durchgeführt werden.

Zunächst muss der Geschwindigkeitsvektor normiert werden.

$$x_F = \frac{v}{|v|} \tag{3.15}$$

Mit der z-Achse des Erdkoordinatensystems z_E kann eine vorläufige y-Achse gebildet werden.

$$y_1 = -z_E \times v_n \tag{3.16}$$

Nun muss die vorläufige z-Achse gebildet werden.

$$z_1 = -\left(y_1 \times v_n\right) \tag{3.17}$$

Die y_1- Achse befindet sich jetzt noch in der x_E y_E-Ebene.

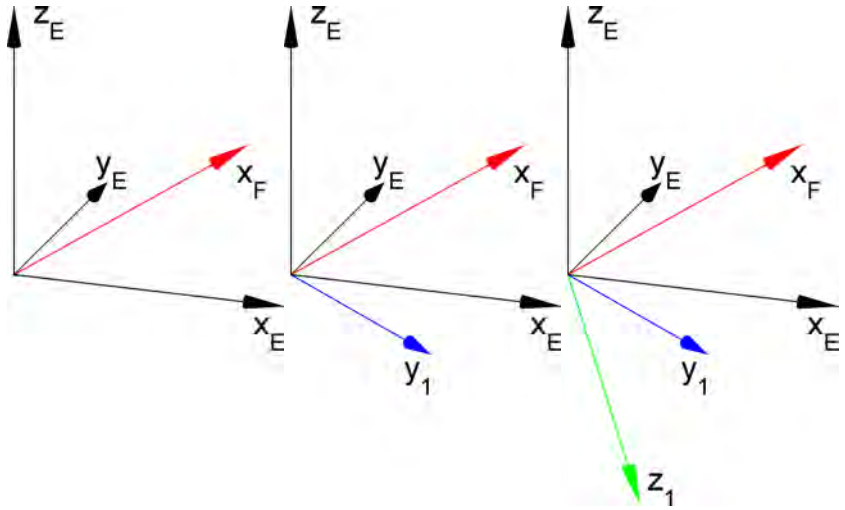


Abb. 3.6: Drehmatrix Schritt 1

Damit die Transformationsmatrix noch eine Drehung um die x_F-Achse beinhaltet, muss zunächst das Verhältniss zwischen Beschleunigung und Erdanziehung gebildet werden. Das Verhältniss gibt den Grad des Rollens an. Aufgrund der bisherigen Drehung ist die Annahme nur gültig, wenn sich der Beschleunigungsvektor in dem vorläufigen Flugzeugkoordinatensystem befindet. Die Beschleunigung muss nämlich vorzeichenbehaftet bleiben, da das Vorzeichen die Richtung des Rollen angibt. Die Drehung kann mit der vorläufigen Transformationsmatrix geschehen.

$$a = \begin{pmatrix} x_F(1,1) & y_1(1,2) & z_1(1,3) \\ x_F(2,1) & y_1(2,2) & z_1(2,3) \\ x_F(3,1) & y_1(3,2) & z_1(3,3) \end{pmatrix} \cdot \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \tag{3.18}$$

Der Grad des Rollens wird nun wie in Abbildung 3.7 bestimmt. Der negative a_y Anteil wird in Richtung y_1 projiziert und der daraus entstehende Vektor mit der vorläufigen z_1-Achse ergibt die z-Achse des Flugzeugkoordinatensystems. Es wird nur der a_y Anteil der Beschleunigung verwendet, da dieser entgegen der Zentrifugalkraft, die auf das Flugzeug wirkt, gerichtet ist.

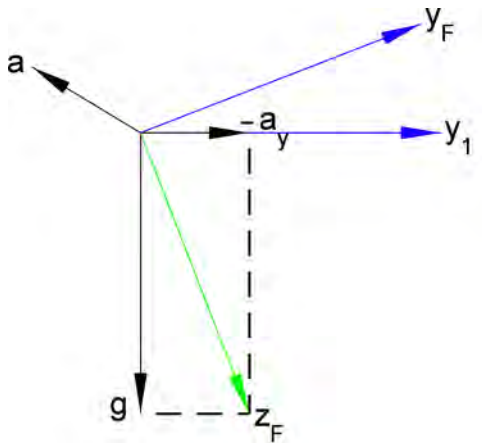


Abb. 3.7: Drehmatrix Schritt 2

Projektion von $-a_y$ Anteil in Richtung y_1.

$$a_{yy_1} = -a_y \cdot y_1 \tag{3.19}$$

Berechnen der z_F- Achse.

$$z_F = z_1 + a_{yy_1} \tag{3.20}$$

Als letzer Schritt muss die y_1- Achse auf die neu entstanden Achsen orthogonal gerichtet werden.

$$y_F = z_F \times x_F \tag{3.21}$$

Die Drehmatrix ist nun vollständig definiert und beschreibt die Drehung von dem Erdkoordinatensystem in das Flugzeugkoordinatensystem eindeutig.

$$R = \begin{pmatrix} x_F(1,1) & y_F(1,2) & z_F(1,3) \\ x_F(2,1) & y_F(2,2) & z_F(2,3) \\ x_F(3,1) & y_F(3,2) & z_F(3,3) \end{pmatrix} \tag{3.22}$$

Anzumerken ist, dass es vor dem Start und nach der Ladung aufgrund der geringen Eigengeschwindigkeit des Flugzeugs und der Positionsungenauigkeit des GPS Empfängers zu Fehlern kommt. Das Flugzeug bewegt sich eigentlich nicht, jedoch können die Positionsangaben des GPS Gerätes zwischen 1 bis ca. 5 Meter streuen und deshalb ändert der Vektor der Geschwindigkeit die Richtung unkontrolliert. Es handelt sich hierbei um keinen Berechnungsfehler im Algorythmus, es sind die Eingangsgrößen (RO-Bert-Datensatz) die den Fehler verursachen. Der Algorythmus wurde mit einer Flugbahn die einer "8" gleicht geprüft[3]. Bei diesem Test konnte festgstellt werden, dass die Bestimmung der Drehmatrix unter den gegebenen Annahmen stimmt.

3.2.4 Eulerwinkel

Die Eulerwinkel, welche die Drehung um x- y- und z-Achse beschreiben, müssen vor der eigentlichen Berechnung der Winkelgeschwindigkeit bestimmt werden. Die drei Winkel können aus der Drehmatrix berechnet werden, jedoch wird diese Lösung nicht eindeutig sein, da beispielsweise $sin(w) = sin(\pi - w) = \frac{\sqrt{2}}{2}$ ist.
Es ist notwendig, einen Definitionsbereich zu bestimmen, um eine Eindeutigkeit der Winkel zu erhalten. Durch die Annahme, dass das Flugzeug keine akrobatischen Manöver ausführt, kann der Drehbereich des Nickens - $R_y(\Theta)$ eingeschränkt werden. Es gilt: $-\frac{\pi}{2} \leq \Theta \leq \frac{\pi}{2}$
Aus der Drehmatrix können somit genau die drei eindeutigen Eulerwinkel Φ, Θ und Ψ bestimmt werden. [Sla99]
In Matlab wird die Berechnung der Eulerwinkel, Winkelgeschwindigkeit und Winkelbeschleunigung mit der Funktion `winkel_v_a` durchgeführt.

$$\begin{aligned} \Theta &= asin\left(r_{3,1}\right) \\ \Phi &= atan2\left(\frac{r_{3,2}}{cos(\Theta)}, \frac{r_{3,3}}{cos(\Theta)}\right) \\ \Psi &= atan2\left(\frac{r_{2,1}}{cos(\Theta)}, \frac{r_{1,1}}{cos(\Theta)}\right) \end{aligned} \tag{3.23}$$

Im Falle des Gimbal Lock, bei dem durch die zweite Drehung die Drehachsen der ersten und dritten Achse kollinear werden, ist die Berechnung nach Formel 3.23 nicht mehr gültig. Der Gimbal Lock tritt bei $r_{3,1} = \pm 1$ bzw. $\Theta = \pm\frac{\pi}{2}$ auf. In diesem Fall würde durch $cos(\pm\frac{\pi}{2}) = 0$ geteilt werden, was nicht zulässig ist. Nach Konvention wird bei dem Auftreten dieses mathematischen Sonderfalles $\Theta = 0$ gesetzt.
Für $\Theta = \frac{\pi}{2}$:

$$\begin{aligned} \Phi &= 0 \\ \Theta &= \frac{\pi}{2} \\ \Psi &= \Theta + atan2\left(r_{1,2}, r_{1,3}\right) \end{aligned} \tag{3.24}$$

[3] Video unter *:\Diplomarbeit\Videos\Drehmatrix_Video.avi

Für $\Theta = -\frac{\pi}{2}$:

$$\begin{aligned} \Phi &= 0 \\ \Theta &= -\frac{\pi}{2} \\ \Psi &= -\Theta + atan2\left(-r_{1,2}, -r_{1,3}\right) \end{aligned} \tag{3.25}$$

3.2.5 Winkelgeschwindigkeit

Die Winkelgeschwindigkeit beschreibt die Änderung des Drehwinkels um seine Drehachse in einem Zeitabschnitt. Es werden die Drehraten um jede Flugzeugachse bestimmt. Bei der Berechnung der Winkeldifferenz muss aufgrund des Definitionsbereiches von atan2 bei einer Drehung von Quadrant 2 nach Quadrant 3, 2π addiert und bei einer Drehung von Quadrant 3 nach Quadrant 2, 2π subtrahiert werden.

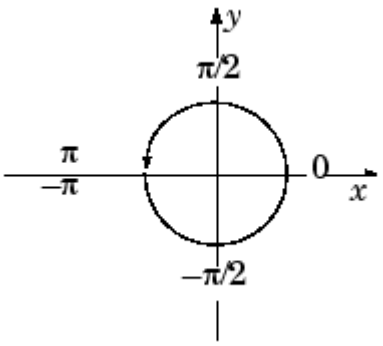


Abb. 3.8: Definitionsbereich atan2
Quelle: MATLAB

```
%
%Beispiel Winkeldifferenz
%
%Winkel 1
w1=160;
%Winkel 2
w2=-170;
%Drehung von dem II in den III Quadranten
%positiver Drehsinn
diffWinkel=w2-w1
if diffWinkel< -180
    diffWinkel=diffWinkel+360
elseif diffWinkel>180
    diffWinkel=diffWinkel-360
end
```

Die Drehrate um die x-Achse wird gemäß Gleichung 3.26, um die y-Achse gemäß Gleichung 3.27 und um die z-Achse gemäß Gleichung 3.28 bestimmt.

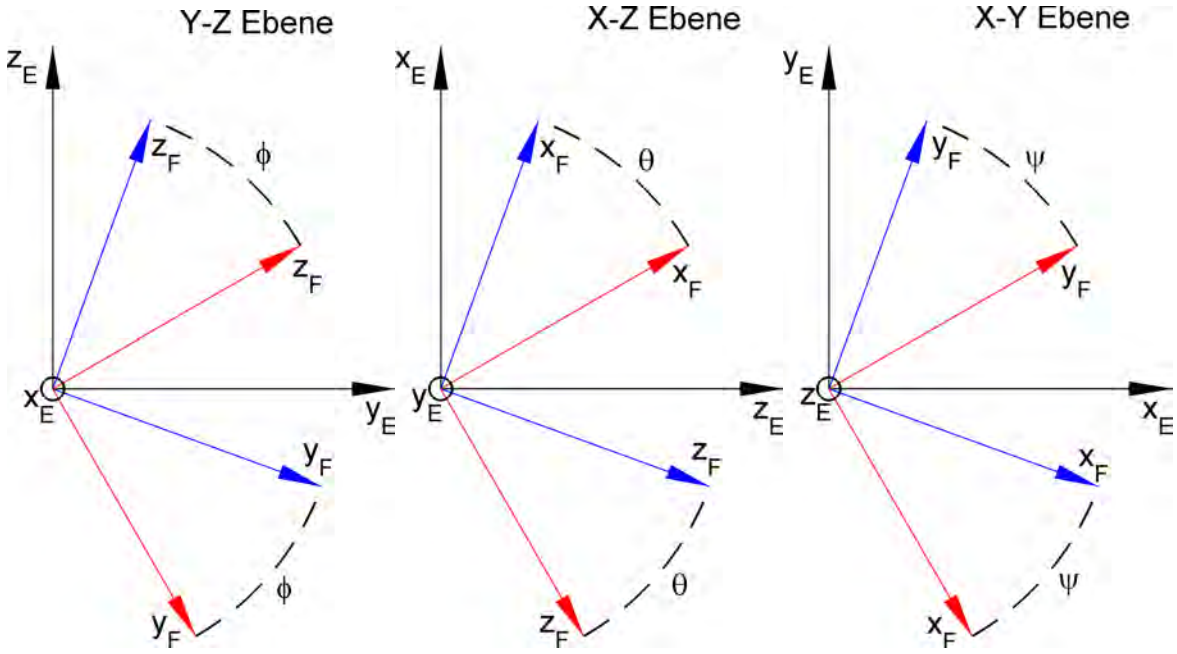


Abb. 3.9: Drehrate um x-Achse, y-Achse und z-Achse

$$\omega_x(t) = \frac{\Phi(t_{n+1}) - \Phi(t_n)}{t_{n+1} - t_n} = \frac{\Delta\Phi}{0.1} \left[\frac{^\circ}{s}\right] \tag{3.26}$$

$$\omega_y(t) = \frac{\Theta(t_{n+1}) - \Theta(t_n)}{t_{n+1} - t_n} = \frac{\Delta\Theta}{0.1} \left[\frac{^\circ}{s}\right] \tag{3.27}$$

$$\omega_z(t) = \frac{\Psi(t_{n+1}) - \Psi(t_n)}{t_{n+1} - t_n} = \frac{\Delta\Psi}{0.1} \left[\frac{^\circ}{s}\right] \tag{3.28}$$

$$\omega = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{3.29}$$

$$|\omega| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \left[\frac{^\circ}{s}\right] \tag{3.30}$$

3.2.6 Winkelbeschleunigung

Die Winkelbeschleunigung beschreibt die Änderung der Winkelgeschwindigkeit in einem Zeitabschnitt. Die Berechnung erfolgt analog zur Beschleunigung.

$$\alpha(t) = \frac{\omega(t_{n+1}) - \omega(t_n)}{t_{n+1} - t_n} = \frac{\Delta\omega}{\Delta t} \left[\frac{^\circ}{s^2}\right] \tag{3.31}$$

$$\begin{pmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{pmatrix} = \frac{\begin{pmatrix} \omega_x(t_{n+1}) \\ \omega_y(t_{n+1}) \\ \omega_z(t_{n+1}) \end{pmatrix} - \begin{pmatrix} \omega_x(t_n) \\ \omega_y(t_n) \\ \omega_z(t_n) \end{pmatrix}}{0.1} \left[\frac{^\circ}{s^2}\right] \tag{3.32}$$

$$|\alpha| = \sqrt{\alpha_x^2 + \alpha_y^2 + \alpha_z^2} \left[\frac{^\circ}{s^2}\right] \tag{3.33}$$

3.2.7 Beschleunigungssensor

Ein Beschleunigungssenor misst die Beschleunigung aufgrund der Trägheitskraft, die auf eine Testmasse wirkt. In Flugzeugen werden Beschleunigungssensoren häufig in Verbindung mit einem Gyroskop zur eindeutigen Lagebestimmung eingesetz. Der Beschleunigungssensor misst die ortsabhängige Beschleunigung an einem Flugzeug, d.h. die Beschleunigung, die zusätzlich zu der Beschleunigung des Flugzeugneutralpunktes auftritt. Die zusätzliche Beschleunigungskomponente setzt sich aus dem Abstand r_i zu dem Neutralpunkt, der Drehrate und Drehbeschleunigung zusammen. Die allgemeine Berechnung erfolgt gemäß Formel 3.34 [Zen11]. In Matlab kann die Berechnung mit der Funktion `Beschlunigungssensoren` durchgeführt werden.

$$\begin{aligned} a_{Sensor(i)} &= a + \tilde{\omega}^2 \cdot r_i + \tilde{\alpha} \cdot r_i \left[\frac{^\circ}{s^2}\right] \\ &= \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} + \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}^2 \cdot r_i + \begin{pmatrix} 0 & -\alpha_z & \alpha_y \\ \alpha_z & 0 & -\alpha_x \\ -\alpha_y & \alpha_x & 0 \end{pmatrix} \cdot r_i \left[\frac{^\circ}{s^2}\right] \end{aligned} \tag{3.34}$$

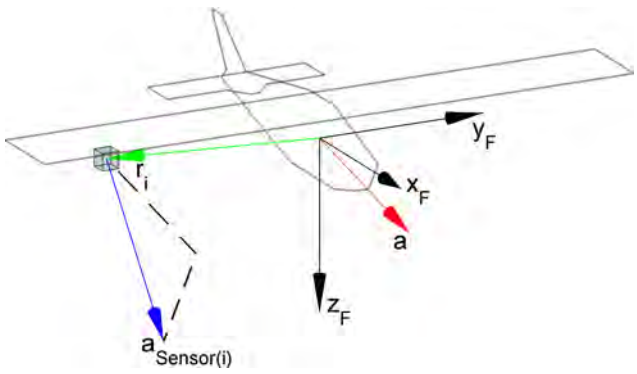


Abb. 3.10: Beschleunigungssensor

3.3 meteorologische Signale

3.3.1 Messwerte DWD

Die Grundlage der Berechnung der Luftfeuchtigkeit und Lufttemperatur sind Messwerte von 13 Bodenstationen des Deutschen Wetter Dienst. Die Messwerte des Jahres 2009 haben eine stündliche Auflösung. Anhand dieser Messwerte werden die Werte der Bodenspur des Flugzeugs zeit- und ortsabhängig interpoliert (Abbildung 3.12). In einem weitern Schritt wird dann die Veränderung mit zunehmender Höhe der interpolierten Werte berechnet um die Flugbahn beschreiben zu können. Zunächst muss dem Zeitvektor ein Datum zugeteilt werden, dies kann mit dier Funktion `Datum_Time` geschehen. Der Zeitvektor als String hat folgendes Format `'dd/mm HH:MM:SS:FFF'` und gibt den genauen Zeitpunkt der Interpolation an. Nun kann abhängig von der Zeit zu jeder Station der entsprechende Luftfeuchtigkeits- und Lufttemperaturwert interpoliert werden. Es besteht jedoch auch die Möglichkeit zeitunabhängig zu Interpolieren (Handbuch 2.3.7). Mit einem modifizierten 4-Punkt Newton Interpolationsalgorythmus[4] werden nun die ortsabhängigen Werte der Bodenspur zwischen den Messstationen im Raum interpoliert. Dieser Algorythmus wurde gewählt, da sich herausstellte, dass die interne Matlab Funktion `interp3` für eine Interpolation dieser Art äußerst ungeeignet ist. Es wird zwischen den vier Punkten mit dem geringsten Abstand zum gesuchten Punkt interpoliert.

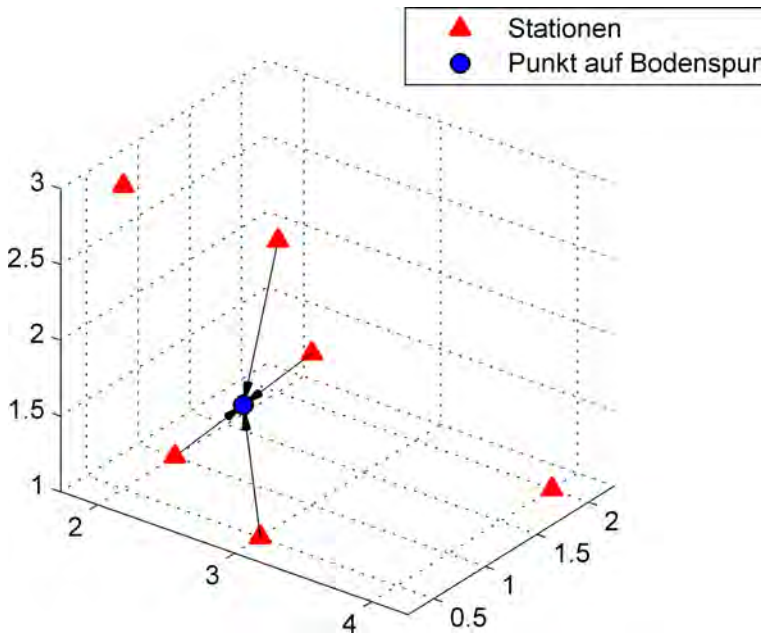


Abb. 3.11: 4 Punkt Newton-Interpolation

In Matlab lässt sich der Interpolationsvorgang mit den Funktionen `interp_Temp_Luftf` zeitabhängig und `interp_Temp_Luftf2` zeitunabhängig Realisieren.

[4] `NewtFit`, Per Sundqvist, Gothenburg University 2005

3.3.2 Lufttemperatur

Die Änderung der Lufttemperatur mit der Höhe wird mit dem in der Meteorologie üblichen Temperaturgradienten berechnet, dieser Beträgt $-\frac{0.65K}{100m}$. Zuerst muss die Höhendiffernz zwischen Bodenspur und Flugbahn bestimmt werden, dann kann die Abnahme der Temperatur über Temperaturgradient und Höhendifferenz berechnet werden. In Matlab kann die Berechnung mit der Funktion `Temperatur_Flug` durchgeführt werden.

$$\Delta z = z_{Flugbahn} - z_{Bodenspur} \quad [m] \tag{3.35}$$

$$T_{Flugbahn} = T_{Bodenspur} + \left(\Delta z \cdot \frac{-0.65}{100}\right) \quad [°C] \tag{3.36}$$

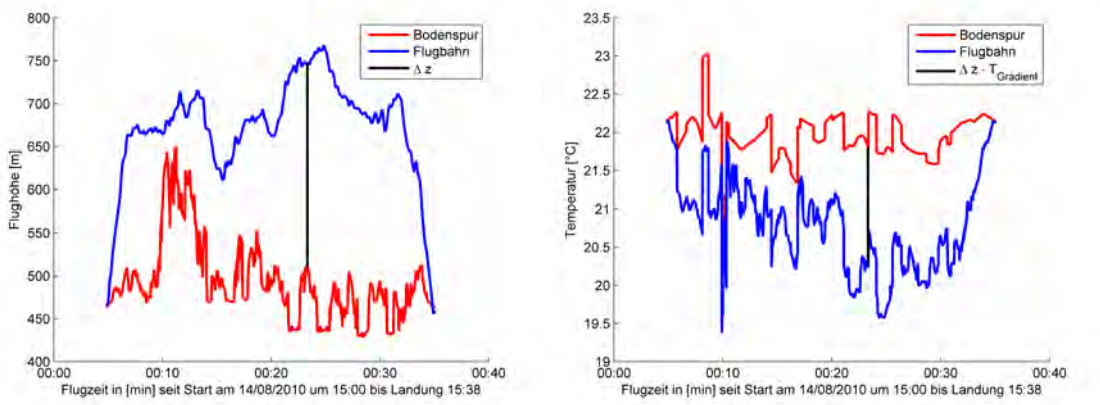


Abb. 3.12: Bestimmung der Lufttemperatur

3.3.3 Luftfeuchte

Die Luftfeuchtigkeit beschreibt den Gehalt der Luft an Wasserdampf. Bei der Berechnung der Luftfeuchtigkeit der Flugbahn muss die Wetterlage berücksichtigt werden, bei klaren Himmel sinkt die Luftfeuchte mit zunehmender Höhe fast linear, sind jedoch Wolken am Himmel ist das nicht der Fall. Bei wolkenbedecktem Himmel steigt die Luftfeuchtigkeit mit Näherung zur Wolke bis auf 100 % an. Um diesen Sachverhalt möglichst genau Simulieren zu können muss zunächst gewählt werden ob der Flug bei wolkenbedecktem oder klarem Himmel erfolgt. Die lineare Abnahme wird gemäß Formel ... berechnet (`Temperatur_Flug`).

$$r_{Flugbahn} = r_{Bodenspur} - \frac{r_{Bodenspur}}{1000} \cdot \Delta z \quad [\%] \tag{3.37}$$

Bei bedecktem Himmel muss zunächst die Luftfeuchte der Bodenspur betrachtet werden, sie gibt Auskunft darüber, wie sich die Luftfeuchte mit der Höhe verändert. Anhand von früheren Messerfassungen wurden verschiedene Szenarien analysiert und bewertet. Anhand der Bewertung konnten sieben verschiedenen Koeffizentenpaare für die Funktion `polyval` bestimmt werden. Jedes Koeffizentenpaar beschreibt die Zunahme der Luftfeuchte abhänging von der Luftfeuchte

der Bodenspur anders. Durch die Koeffizenten p lässt sich eine höhenabhängige Funktion zweiten Grades definieren, die die Luftfeuchtigkeit an der Flugbahnpostion beschreibt.

$$r_{Flugbahn}(z) = p_1 z^2 + p_2 z \quad [\%] \tag{3.38}$$

In Matlab wird die Berechnung über die Funktion `Luftfeuchte_Flug` realisiert. Da es sich um ein Polynom handelt kann es bei großer Höhe vorkommen, dass die Luftfeuchtigkeit über 100 % steigt. Innerhalb einer Wolke herrscht ca.100 %, deshalb werden alle Werte die größer als 100 % sind, auf 100 % gesetzt. Diese Vorgehensweise beeinträchtig die Genauigkeit nicht, da sich Wolken über einen Höhenbereich von ca. 1500m - 15000m erstrecken.

```
%
%Berechnung Luftfeuchte Flugbahn
%
i=length(z_Flug);
%Unterscheidung der Koeffizenten
if r_Boden(1,1)<55
    load LF55.mat
elseif r_Boden(1,1)≤60
    load LF60.mat
elseif r_Boden(1,1)≤70
    load LF70.mat
elseif r_Boden(1,1)≤80
    load LF80.mat
elseif r_Boden(1,1)≤85
    load LF85.mat
elseif r_Boden(1,1)≤95
    load LF95.mat
elseif r_Boden(1,1)≤100
    load LF100.mat
end
r_Flug=polyval(pk,z_Flug);
for n=1:i
    if r_Flug(n,1)≥100;
        r_Flug(n,1)=100;
    end
end
```

3.3.4 Wind

Unter Wind versteht man in der Meteorologie eine zielgerichtete Luftbewegung in der Atmosphäre. Wind entsteht zwischen Luftschichten mit unterschiedlichen Luftdrücken, dabei fliegen Luftteilchen so lange von dem Hochdruckgebiet zu dem Tiefdruckgebiet bis der gleiche Luftdruck

zwischen den Gebieten herrscht. Da es sich bei der Luftbewegung um ein sehr komplexes Thema handelt, wird eine vereinfachtes Modell verwendet, bei dem Luftbewegung nur in der xy-Ebene stattfindet. Die Generierung des Windes erfolgt nach den Vorgaben von Windgeschwindigkeit und Windrichtung, diese sind in Zeitmarken gespeichert. Ein Zustand hält so lange an, bis die folgende Zeitmarke erreicht wird (Handbuch 2.3.5). Die Flugbahn der Hagelwolke wird durch die Eingaben ebenso beeinflusst wie der dynamische Druck. Die Windrichtung wird in Grad eingegeben, daraus lässt sich die Himmelsrichtung aus eine Windrose ableiten.

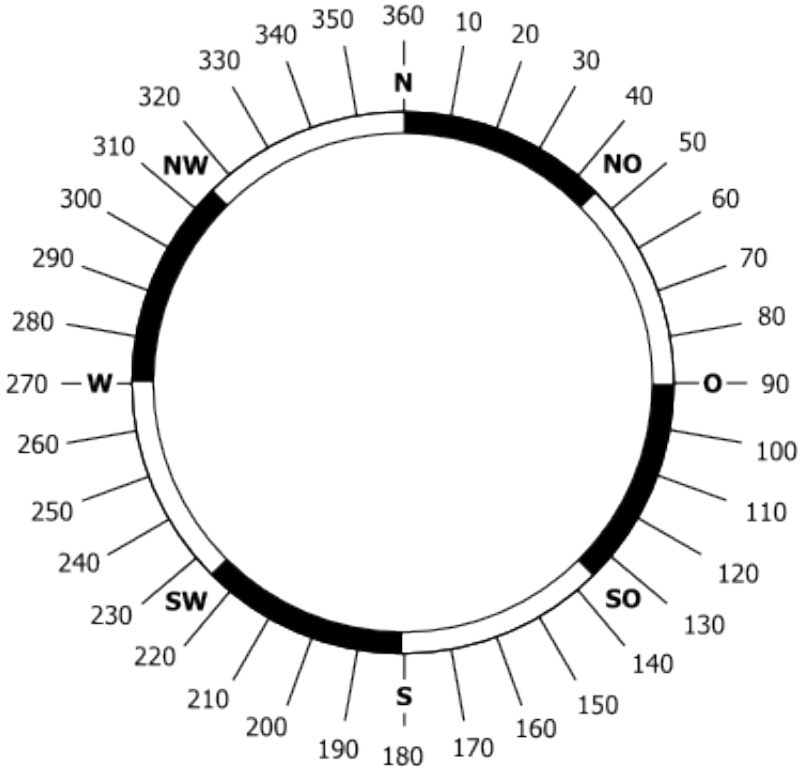


Abb. 3.13: Windrose
Quelle: http://www.pzgrenbtl417.eu/

Für die Berechnung der vektoriellen Größe Wind muss zunächst eine Drehung der normierten y-Achse des Erdkoordinatensystem um die z-Achse mit dem Richtungswinkel α ausgeführt werden. Der Winkel wird zuvor mit 180 Grad addiert, da die Windrichtung die Richtung angibt, aus der der Wind weht und die Wolkenbewegung entgegengesetzt stattfindet.

$$RV_{Wind} = R_z(180 + \alpha) \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \tag{3.39}$$

Wird der Richtungsvektor nun mit der Windgeschwindigkeit multipliziert, enthält die vektorielle

Größe Wind, Richtung und Geschwindigkeit im Erdfestenkoordinatensystem.

$$\begin{pmatrix} Wind_x \\ Wind_y \\ Wind_z \end{pmatrix} = RV_{Wind} \cdot v_{Wind} \tag{3.40}$$

Die Flugbahn der Wolke wird über einen Startpunkt und die addierten Windgeschwindigkeitsvektoren definiert.[5]

3.3.5 Luftdruck

3.3.5.1 statisch

Aufgrund der Schwerkraft übt Luft eine Kraft auf eine Fläche aus, den hydrostatischen Druck, dieser ist abhängig von der Luftmenge oberhalb der betrachteten Fläche. Der Luftdruck nimmt deshalb mit zunehmender Höhe über der Erdoberfläche ab. Die Abnahme des Luftdrucks mit der Höhe lässt sich über die barometrische Höhenformel[Mal06] berechnen. Diese Formel berechnet den Druckunterschied zwischen zwei Höhenschichten abhängig von Temperatur, Höhe über Normalnull und Luftfeuchte. Als Bezugsniveau werden die Werte der internationalen Standartatmosphäre[6] auf Meeresniveau verwendet, Temperatur $T_o = 15\,°C$ und Luftdruck $p_0 = 1013.25\,mbar$. Zunächst muss die mittlere virtuelle Temperatur $\bar{T_v}$ zwischen den Höhenschichten bestimmt werden, dies ist die Temperatur bei der trockene Luft die selbe Dichte wie feuchte Luft hat. Dadurch ist es möglich, mit der Gaskonstante von trockener Luft R_L zu rechnen. Zur Bestimmung der mittleren virtuellen Temperatur wird die spezifische Feuchte s, aus dem Verhältnis zwischen Dampfdruck e und Luftdruck p_0 berechnet. Der Dampfdruck wird aus dem Sättigungsdruck e_s, welcher näherungsweise mit der Magnus-Formel[7] bestimmt werden kann und der Luftfeuchte r berechnet.

$$e = \frac{es \cdot r}{100} \quad [mbar] \tag{3.41}$$

$$s = 0.662 \cdot \frac{e}{p_0} \tag{3.42}$$

$$\bar{T_v} = \frac{T_{Flugbahn} + T_0}{2} \cdot (1 + 0.61s) \tag{3.43}$$

Der Luftdruck an einer Flugzeugposition kann nun gemäß der barometrischen Höhenformel berrechnet werden. (`Luftdruck`)

$$p_{statisch} = p_0 \cdot e^{-\frac{gz}{R_L \bar{T_v}}} \quad [mbar] \tag{3.44}$$

[5] Video unter: *:\Diplomarbeit\Videos\Simulation_Wolke.avi
[6] Internationale Zivilluftfahrtorganisation, 1952
[7] Heinrich Gustav Magnus, 1844

3.3.5.2 dynamisch

Der dynamische Luftdruck entsteht, wenn die kinematische Energie des Windes auf eine Fläche trifft. Aus der Luftdichte ρ und Windgeschwindigkeit v lässt sich der dynamische Luftdruck berechen.

$$p_{dynamisch} = \frac{1}{2} \cdot \rho \cdot v^2 \quad [mbar] \tag{3.45}$$

Für die Berechnung des dynamischen Luftdrucks am Flugzeug muss zunächst der True Airspeed und die veränderbare Luftdichte bestimmt werden. True Airspeed ist die Geschwindigkeit des Flugzeugs relativ zur unbeeinflussten Luft. Die Luftdichte verändert sich abhängig von der Flugzeugposition.

$$v_{TA} = v_{Flugzeug} - v_{Wind} \quad \left[\frac{m}{s}\right] \tag{3.46}$$

$$\rho_{Flugbahn} = \frac{p_{statisch}}{R_L \cdot \bar{T}_v} \quad \left[\frac{kg}{m^3}\right] \tag{3.47}$$

Für den dynamischen Luftdruck am Flugzeug ergibt sich nun: (`dynamischer_Druck`)

$$p_{dynamisch} = \frac{1}{2} \cdot \rho_{Flugbahn} \cdot v_{TA}^2 \quad [mbar] \tag{3.48}$$

3.4 geophysikalische Signale

3.4.1 magnetische Feldstärke

Die Berechnung der magnetischen Feldstärke wird anhand des Modells "International Geomagnetic Reference Field" (IGRF-11)[8] durchgeführt. Das IGRF-11 ist weltweit als Standardmodell zur Beschreibung des Erdmagnetfeld anerkannt. Die "International Association of Geomagnetism and Aeronomy" stellt ein Fortran Programm zur Berechnung des Magnetfeldes zur Verfügung. Dieses Programm wurde umgeschrieben, so dass es in Matlab als Funktion vorliegt. Mit `igrf11syn`[9] kann abhängig von geographischer Breite, geographischer Länge und Höhe über Normal Null die vektorielle Größe der magnetischen Feldstärke berechnet werden.
Damit gewährleistet wird, dass die Berechnungen auch der Realität entsprechen, wurde ein Vergleich mit dem jährlichen Magnetogramm von 2011 des Geophysical Observatory Fürstenfeldbruck durchgeführt.

```
%
%Berechnung der magnetischen Feldstärke – Fürstenfeldbruck
%
%geo. Breite [dezimal Grad]
Lat=48.165
%geo. Länge [dezimal Grad]
Lon=11.277
%Höhe über NN [km]
z=0.570
%
[B_Feld] = Magnetisches_Feld(48.165, 11.277,0.570,gh)

B_Feld =
        x: 780.5654       %
        y: 2.0962e+004    % H= Horizontalintensität [nano Tesla]
        z: 4.3310e+004    % Z= Vertikalintensität   [nano Tesla]
        Vec: 4.8122e+004  % F= Magnetfeldstärke     [nano Tesla]
```

[8] http://www.ngdc.noaa.gov/IAGA/vmod/
[9] http://www.mathworks.com/matlabcentral/fileexchange/28874-igrf-magnetic-field

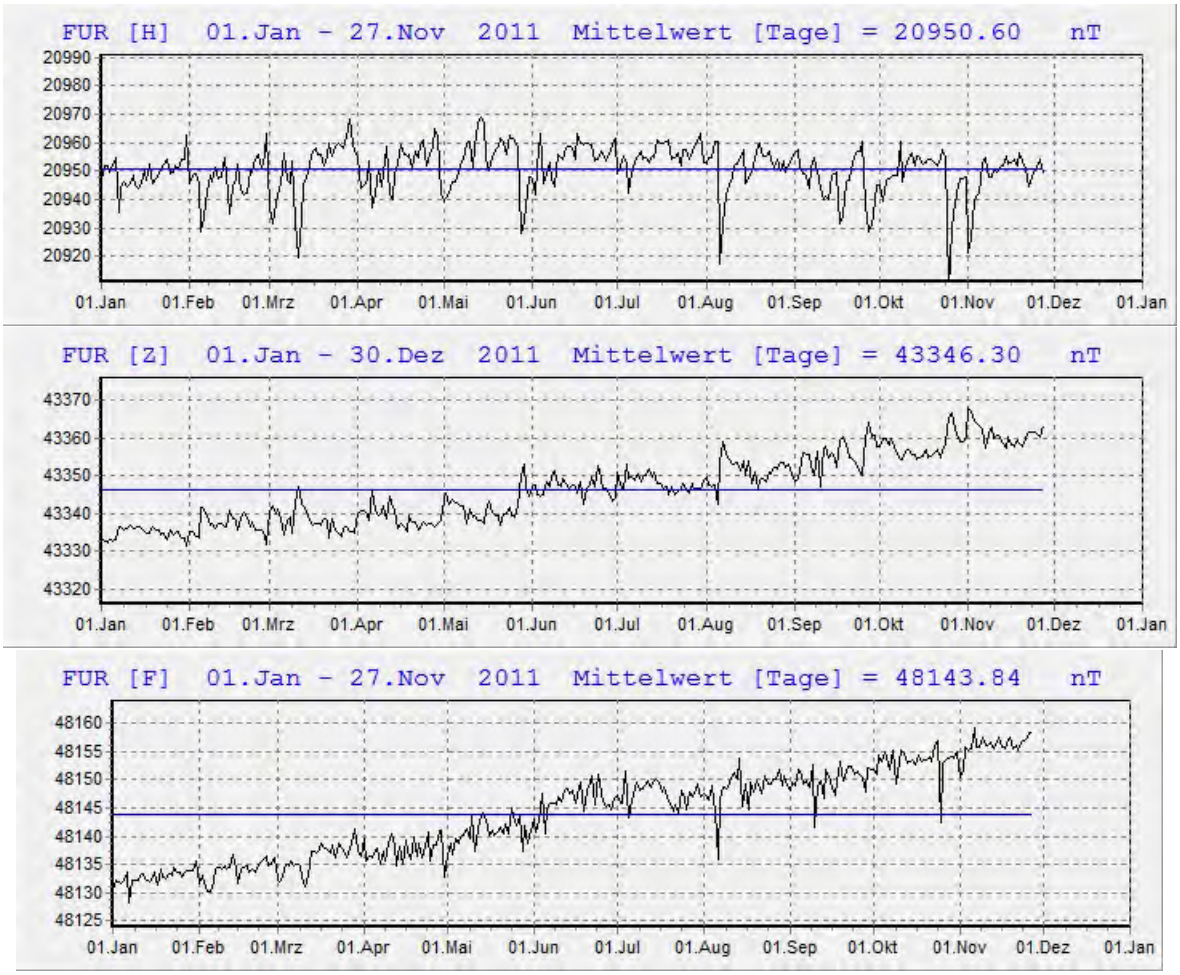


Abb. 3.14: Jährliches Magnetogramm - Fürstenfeldbruck
Quelle: http://www.geophysik.lmu.de/observatory/geomagnetism/

Der Vergleich zwischen IGRF-11 Modell und Messung bestätigt, dass die Berechnung durchaus als realitätsnah angesehen werden kann, da die Abweichung zum Jahresmittelwert für Horizontalintensität und Vertikalintensität nur ca. 50nT beträgt.

3.4.2 elektrische Feldstärke

Die elektrische Feldstärke beschreibt die Fähigkeit des elektrischen Feldes, Kraft auf eine Ladung auszuüben. Die Kraft wird in einem Punkt im Raum durch einen Vektor definiert. Die Feldstärke der bodennahen Atmosphäre wird weitgehendst durch die Ionisation der Erdoberfläche bestimmt. Für die Ionisation ist die radioaktive Strahlung der Erdoberfläche, sowie Radon aus den oberen Bodenschichten und dessen Zerfallprodukte, verantwortlich. Es gibt jedoch auch noch andere Komponenten, welche die Ionisation beeinflussen, wie z.B. Umspannwerke und Hochspannungsleitungen. Als Mittelwert der el. Feldstärke wird an der Erdoberfläche 130 $\frac{V}{m}$ [Leu04] angegeben. Für die Modellberechnung wird angenommen, dass der Erdkern eine Punktladung ist. Dadurch lässt sich die Formel für die Feldstärke einer Kugel verwenden.

$$E(r) = \frac{Q}{4\pi\epsilon_0 r^2} \quad \left[\frac{V}{m}\right] \tag{3.49}$$

Zuerst muss die Ladung Q_0 des Erdkerns berechnet werden. Durch den Mittelwert der el. Feldstärke $\bar{E}$ auf der Erdoberfläche und dem mittleren Erdradius $\bar{r}$ wird die Punktladung bestimmt.

$$Q_0 = \bar{E} 4\pi\epsilon_0 \bar{r}^2 \quad [C] \tag{3.50}$$

Die Abnahme des Betrages der el. Feldstärke hängt jetzt nur noch mit dem Abstand zum Erdkern zusammen. Der Abstand setzt sich aus dem mittleren Erdradius und der Flughöhe zusammen. Aufgrund des großen Radius wurde noch ein Faktor ^{k} eingeführt, der die Auswirkung der Flughöhe z verstärkt.

$$|E| = \frac{Q_0}{4\pi\epsilon_0 \left(\bar{r} + z^k\right)^2} \quad \left[\frac{V}{m}\right] \tag{3.51}$$

Die Richtung der elektrischen Feldstärke muss noch berechnet werden. Da die Erde negativ und die Atmosphäre positiv geladen ist, kann eine grobe Richtung vorgegeben werden. Für eine genaue Ausrichtung muss ein Punkt auf der Erdoberfläche bestimmt werden über dem sich das Flugzeug befindet. Der Vektor dieses Punktes zu dem Erdkern gibt die genau Ausrichtung an. Der Punkt auf dem Erdellipsoiden wird mit a = große Halbachse, e = erste elliptische Exzentrizität, lat = Breitengrad und lon = Längengrad berechnet. [Sei06]

$$\begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = \begin{pmatrix} \frac{a\,cos(lat)\,cos(lon)}{\sqrt{1-e^2\,sin^2(lat)}} \\ \frac{a\,cos(lat)\,sin(lon)}{\sqrt{1-e^2\,sin^2(lat)}} \\ \frac{a\left(1-e^2\right)sin(lat)}{\sqrt{1-e^2\,sin^2(lat)}} \end{pmatrix} \tag{3.52}$$

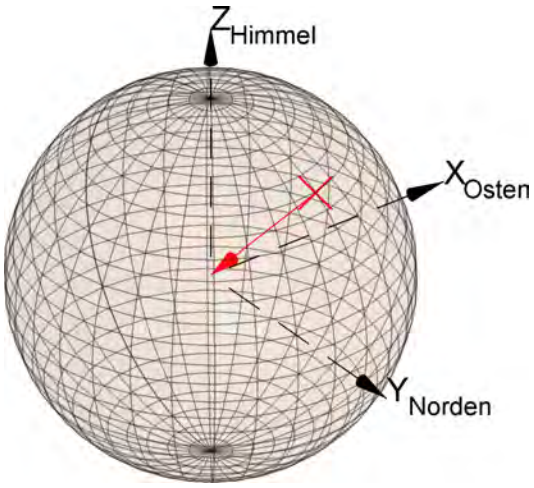


Abb. 3.15: Ausrichtung der el. Feldstärke

Die vektoriellen Komponenten der el. Feldstärke errechnen sich aus dem normierten Richtungs-

vektor r_E und dem Betrag der Feldstärke.

$$\begin{pmatrix} E_x \\ E_y \\ E_z \end{pmatrix} = r_E \cdot |E| \quad \left[\frac{V}{m}\right] \tag{3.53}$$

3.5 sonstige Signale

3.5.1 Feinstaubdichte

Als Feinstaub bezeichnet man alle Partikel des Schwebstaubes in der Atmosphäre, die einen Durchmesser $\leqq 10\mu m$ haben. Von dieser geometrischen Angabe stammt auch die alternative Bezeichnung von Feinstaub - PM_{10}. Zu den Hauptursachen der Feinstaubemission zählt die Industrie und der Straßenverkehr. Deshalb kann angenommen werden, dass die Feinstaubdichte mit zunehmender Höhe abnimmt, da sich die Emissionsquellen auf der Erdoberfläche befinden. Die Bestimmung der Feinstaubwerte erfolgt analog zur Bestimmung der Lufttemperatur. Zuerst wird die Bodenspur zwischen 4 Messstationen des Landesamtes für Umwelt Bayern interpoliert, danach muss die Abnahme der Feinstaubwerte aufgrund der zunehmenden Höhe berechnet werden. (`PM10_Flug`)

$$PM_{10\ Flugbahn} = PM_{10\ Bodenspur} - \frac{PM_{10\ Bodenspur}}{1000} \cdot \Delta z \quad \left[\frac{\mu g}{m^3}\right] \tag{3.54}$$

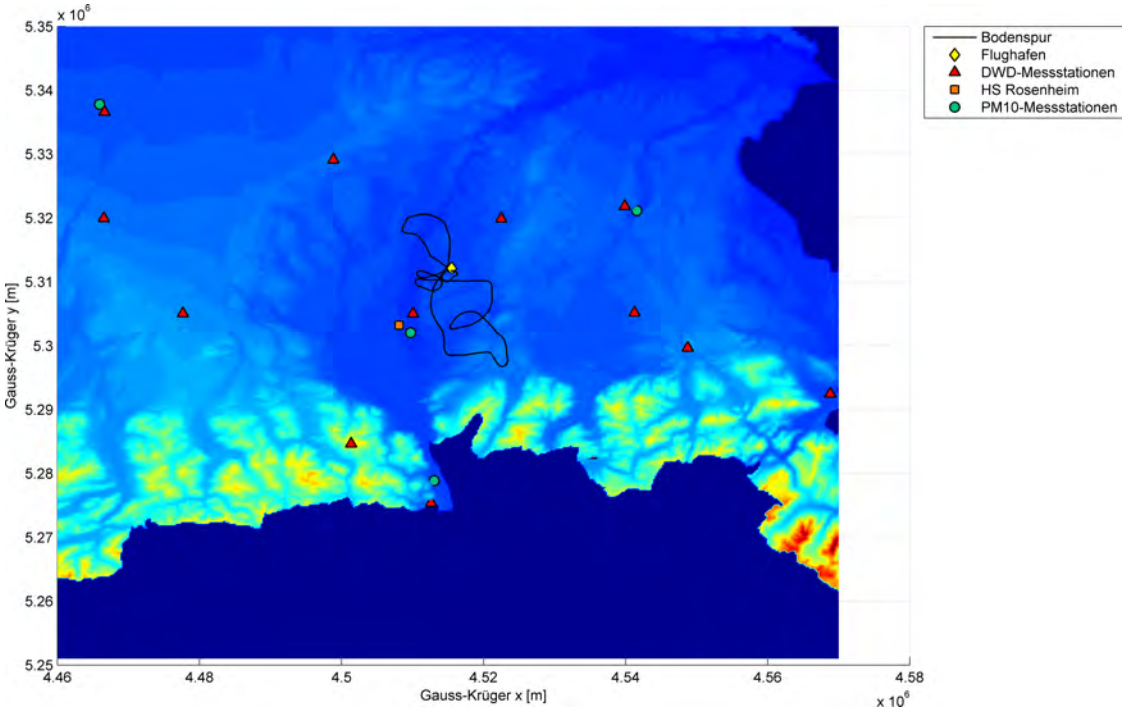


Abb. 3.16: Einsatzgebiet mit Messstationen

3.5.2 AgI-Kanonen

Die Leistungsstärke der zwei Silberjodidkanonen des Flugzeugs wird über eine digitale Steuerung bestimmt. Durch die Eingabe von Bit-Werten und Zeitabschnitten wird der Flugbereich ähnlich wie bei der Generierung des Windes definiert. Es ist anzumerken, dass nicht festgestellt werden konnte, ob die Änderung der Intensitätsstufen der Silberjodidkanonen tatsächlich linear mit der Zunahme der Bit-Werte geschieht, da leider keinerlei Unterlagen bzw. Datenblätter vorhanden sind.

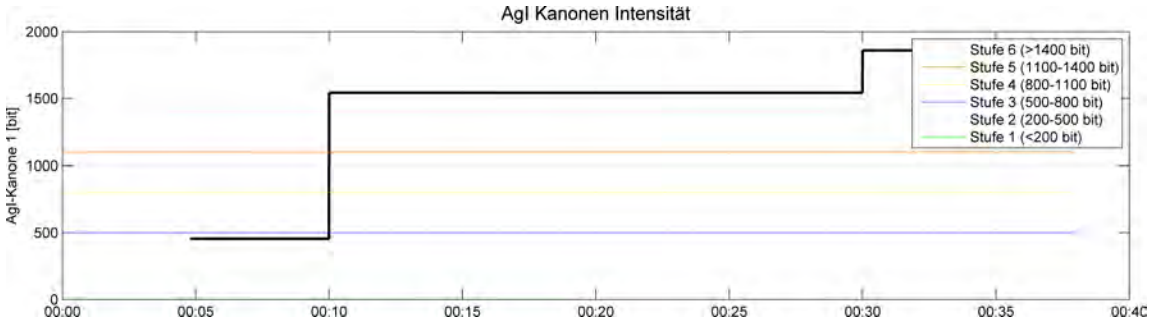


Abb. 3.17: AgI-Intensität

3.5.3 Hagelwolke

Bei der Simulation der Hagelwolke wurde darauf geachtet, dass die Generierung der Wolke anhand eines künstlichen Radarbildes erfolgt. Zunächst ist zu sagen, dass der DWD durch das Wetterradar die Möglichkeit besitzt, Echo-Intensitätsklassen von Wolkenfronten zu bestimmen und die daraus gewonnen Daten in eine CSV-Datei zu speichern. Zum Zeitpunkt der Bearbeitung der Diplomarbeit sind Testdaten eines $200km\ x\ 200km$ großen Gebietes vorhanden, welches das Hageleinsatzgebiet mit einer 1 km Rasterung abdeckt. Die Struktur der CSV-Datei ist eine Matrix mit 2400 Zeilen und 200 Spalten pro Radarbild. Die Radarmessung erfasst das Gebiet in 12 unterschiedlichen Höhenschichten (1km - 12km) in 15 minütigem Takt. Die erste Höhenschicht in der Matrix wäre z.B. `Radarbild(1:200,1:200)`, die zweite `Radarbild(201:400,1:200)` und so weiter. Die Werte der Matrix repräsentieren die Stärke der Echoklasse, 0 - keine Reflektion und 6 - sehr starke Reflektion. Der Importiervorgang nacheinander aufgenommer CSV-Dateien kann mit `DWD_Import` geschehen. Der nächste Schritt besteht darin, die Position und Echoklasse der Matrixwerte richtig zu interpretieren und visualisieren, dies kann mit der Funktion `DWD_Wolke` durchgeführt werden.[10]
Zur Simulation der Bewegung einer Hagelzelle reicht es aus, nur die Hagelzelle selbst darzustellen und nicht ein ganzes Gebiet. Die Echoklassen können genau so wie bei den DWD-Daten je nach Intensität in einer Matrix definiert werden. Die grafische Darstellung erfolgt mit dem Befehl `patch`, dieser ermöglicht es, einfache Würfel zu erstellen. Die Farbgestaltung der Würfel orientiert

[10]Video unter: *:\Diplomarbeit\Videos\Simulation_DWDWolke.avi

sich an den üblichen Farben der unterschiedlichen Echoklassen.

Radar <81x9 double>

	1	2	3	4	5	6	7	8	9
1	1	0	0	0	0	0	0	0	1
2	0	2	0	0	0	0	0	2	0
3	0	0	3	0	0	0	3	0	0
4	0	0	0	0	5	0	0	0	0
5	0	0	0	5	6	5	0	0	0
6	0	0	0	0	5	0	0	0	0
7	0	0	3	0	0	0	3	0	0
8	0	2	0	0	0	0	0	2	0
9	1	0	0	0	0	0	0	0	1

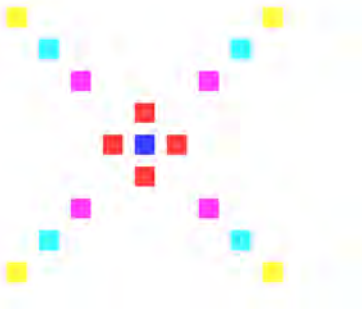


Abb. 3.18: Algorithmus Radardaten - 1 Höhenschicht

Es ist nötig, die erstellten `patch`-Objekte zu bewegen, damit ein Verlauf der Hagelwolke simuliert werden kann. Die `patch`-Objekte, deren Echoklasse nicht Null ist, werden als `Handel`-Objekte gespeichert. Dadurch ist es möglich, die Position jedes einzelnen Würfels durch den Befehl `set(Handel,'xdata',x,'ydata',y,'zdata',z)` im kartesischen Koordinatensystem zu ändern. Bei der Auswertung hat man nun die Möglichkeit, eine Hagelzelle gemäß ihrer Flugbahn über die Landkarte ziehen zu lassen. Dabei richtet sich die Zelle automatisch entsprechend der Windrichtung aus.[11]

[11] Video unter: *:\Diplomarbeit\Videos\Simulation_Wolke.avi

4 Auswertung

Die Auswertung der generierten Daten ist ein wichtiger Bestandteil. Der Mensch kann eine große numerische Datenmenge nur schwer verarbeiten, deshalb wird die komplette Auswertung der Messdaten grafisch durchgeführt. Die Visualisierung der Daten wird durch Diagramme, welche mit dem Befehl `plot` erstellt werden, aber auch durch komplette Simulationsabläufe die Bewegung darstellen, ausgeführt. Durch die Vielzahl der Plots wäre es nicht sinnvoll jeden separat zu erläutern, deshalb wird als exemplarisches Beispiel der Plot der Beschleunigungssensoren gewählt, um die wichtigsten Funktionsweisen zu erklären. Alle Diagramme sind dynamisch programmiert, das heißt, falls veränderliche Größen vorhanden sind, wird dies in einem Plot berücksichtigt.

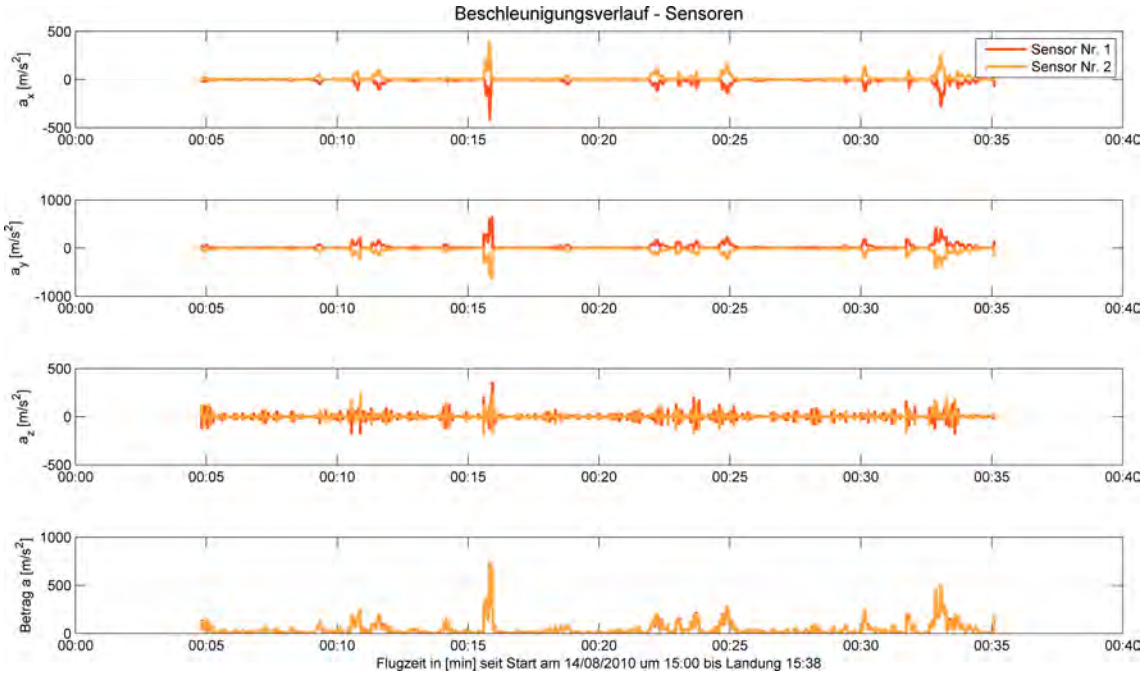


Abb. 4.1: Plot - Beschleunigungssensor

Flugzeit und Datum ändern sich mit jeder Datengenerierung, deshalb setzt sich die x-Achsenbeschriftung des `subplot(4,1,4)` aus verschiedenen Strings zusammen. Datum und Flugzeit sind jeweils in einer Variablen gespeichert und können in die Achsbeschriftung mit eingefügt werden. Die Anzahl der Beschleunigungssensoren kann auch nach Belieben verändert werden, deshalb wurde hier ein Algorithmus in den Plot eingefügt der automatisch erkennt wie viele Sensoren berechnet wurden und teilt ihnen jeweils eine andere Farbe zu. Damit die Übersicht nicht verloren

geht, passt sich die Legende auch automatisch mit an. Durch die Vielzahl der Plots muss eine Möglichkeit gefunden werden, um dem User die Eingabe der verschiedenen Plotbefehle zu ersparen. Die Steuerung über ein GUI bietet sich hierfür am besten an, da das Aufrufen der Plots einfach per Point & Click geschehen kann. Das GUI wird automatisch nach Programmablauf gestartet, jedoch kann es auch manuell mit `Plotten_A`, `Plotten_B` und `Plotten_C` gestartet werden.

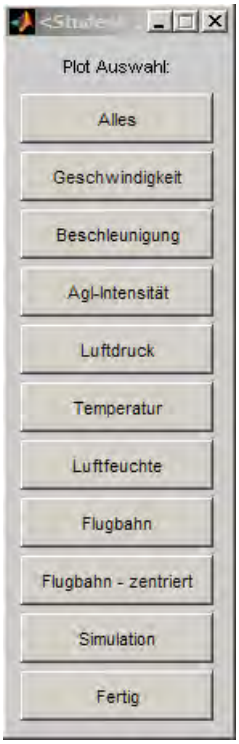


Abb. 4.2: GUI - Ablauf_A

Die Darstellung in Diagrammform eignet sich nicht für alle Messsignale, beispielsweise könnte man die drei Eulerwinkel zwar in einem Diagramm darstellen, jedoch ist die Darstellung des gedrehten Flugzeugs viel anschaulicher und Fehler könnten viel schneller erkannt werden. Als Simulation dargestellt werden die Lage des Flugzeugs, DWD-Wetter Radar, Hagelwolke und die Lage des Flugzeugs mit Flugbahn.

5 Matlab - MySQL

Die Verbindung zwischen MATLAB und MySQL wird durch einen in MATLAB kompilierten C-Code hergestellt. Das mYm-Interface[1] ermöglicht Befehle an eine Datenbank in normaler SQL Syntax zu senden. Um die Kommunikation zwischen MATLAB und Datenbank zu erleichtern, werden zudem die mYm-Utilities[2] verwendet. Bei den mYm-Utilities handelt es sich um Functions, die Befehle in SQL Syntax übersetzen. Der schematische Verbindungsablauf ist in Abbildung 3.1 dargestellt.

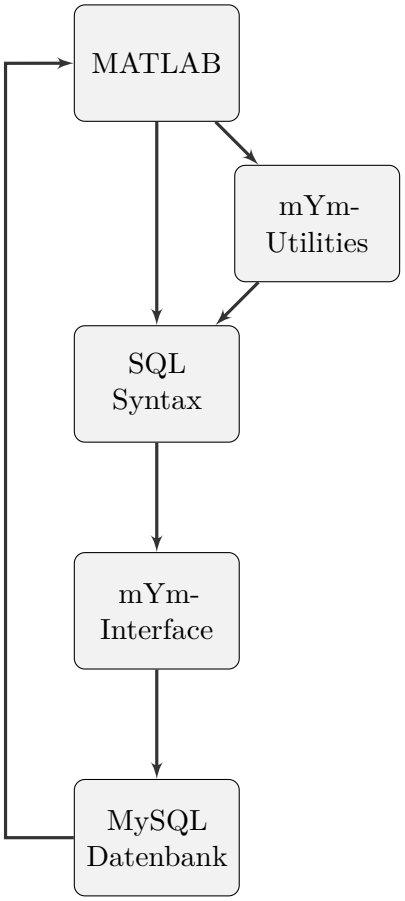


Abb. 5.1: Kommunikation MATLAB-MySQL

[1] mYm v1.0.8, Copyright (C) 2006, Swiss Federal Institute of technology, Lausanne, CH
[2] mYm Utilities, 2007, Dimitri Shvorob

5.1 mYm-Utilities

Bei der Bedienung der mYm-Utilities sind einige grundsätzliche Regeln zu beachten. Es können nur Daten des Datentyps `char` in eine Datenbank übermittelt werden. Zahlenformate müssen folglich in einen Character-String umgewandelt werden. Die übermittelten Zahlen des Datentypes `char` werden automatisch in das definierte Spaltenformat der Datenbank konvertiert, z.B. `double`. Ausserdem sind Inputvariablen, Outputvariablen, Spaltennamen, und Spaltendatentypen in einem jeweils seperaten Cell-Array zu definieren.

5.1.1 Beispiel

Anhand eines Beispiels soll der Mechanismus genauer erläutert werden:
`Leinenbach\MySQL\Beispielprogramm.m`

```
%Beispielprogramm MATLAB–MySQL
%
%write
%
%Server Me–Re–Technick
host='141.60.140.43';
user='leinenbach';
pwd='leinenbach';
%Herstellen Verbindung
myopen(host,user,pwd);
%Prüft die Verbindung
con=myisopen;
if con==1
disp(['Verbindung zum Server: ', host, ' hergestellt']);
else
disp(['Verbindung konnte nicht hergestellt werden']);
end
%Anzeigen allder Datenbanken
dblist
%Öffnet D.B.
dbopen('IdealerSensor_Beispiel')
%Anzeigen aller Tabellen
tblist
%Array der Spaltennamen
Spaltennamen={'Nachname', 'Vorname', 'Geburtsjahr'};
%Array des Datenformates
Datentyp={'varchar(30)', 'varchar(30)', 'double'};
%Erstellen einer Tabelle
%Replace, damit Beispiel reproduzierbar ist
tbadd('Mitarbeiter',Spaltennamen,Datentyp,'replace');
%Inputvariablen
Nachname_sql={'Müller', 'Meier'};
Vorname_sql={'Sepp', 'Hans'};
```

```
Geburtsjahr_sql={'1982', '1945'};
%Übergabearray Definieren
Inputvariablen={'Nachname_sql', 'Vorname_sql', 'Geburtsjahr_sql'};
%Datenübertragung Matlab—MySQL
tbwrite('Mitarbeiter',Spaltennamen,Inputvariablen);
%
%read
%
%Verbindungsaufbau analog zu write
%Workspacevariablen definieren
Outputvariablen={'Nachname_read', 'Vorname_read', 'Geburtsjahr_read'};
%Datenübertragung MySQL—Matlab
%'' = Art der Datenauswahl —> Alle
tbread('Mitarbeiter',Spaltennamen,Outputvariablen,'');
%Verbindung Beenden
myclose;
```

SQL-Abfrageergebnis

Host: localhost
Datenbank: IdealerSensor_Beispiel
Erstellungszeit: 25. November 2011 um 14:42
Erstellt von: phpMyAdmin 3.3.10 / MySQL 5.1.46-log
SQL-Befehl: SELECT * FROM `Mitarbeiter` LIMIT 0, 30 ;
Zeilen: 2

Nachname	Vorname	Geburtsjahr
Müller	Sepp	1982
Meier	Hans	1945

Abb. 5.2: Datenübertragung Beispielprogramm.m

5.1.2 Funktionstabelle

Übersicht aller verfügbaren mYm-Utilities Funktionen.

Tab. 5.1: mYm-Funktionen

Funktion:	Aufgabe:	Syntax:
myisopen	Prüfen einer MySQL Verbindung	`myisopen`
myopen	Herstellen einer MySQL Verbindung	`myopen('Host','User','Passwort');`
myclose	Schließen einer MySQL Verbindung	`myclose`
dblist	Auflisten aller Datenbanken	`dblist`
dbopen	Öffnet Datenbank	`dbopen('Datenbank');`
dbcurr	Zeigt aktuelle Datenbank an	`dbcurr`
dbadd	Erstellen einer Datenbank	`dbadd('Datenbank');`
dbdrop	Löschen einer Datenbank	`dbdrop('Datenbank');`
tblist	Auflisten aller Tabellen	`tblist`
tbadd	Erstellen einer Tabelle	`tbadd('Tabelle',{'Spalten'},...` `{'Datentypen'};`
tbdrop	Löschen einer Tabelle	`tbdrop('Tabelle');`
tbrename	Umbenennen einer Tabelle	`tbrename('Tabelle',...` `'Tabelleneu');`
tbattr	Auflisten Spaltennamen und -typen.	`[Namen, Typen]=tbattr('Tabelle')`
tbsize	Tabellendimension Zeilen x Spalten	`[size]=tbsize('Tabelle');`
tbread	Lesen aus Tabelle	`tbread('Tabelle',{'Spalten'},...` `{'Outputvariablen'});`
tbwrite	Schreiben in Tabelle	`tbwrite('Tabelle',{'Spalten'},...` `{'Inputvariablen'});`
mym	Sendet SQL Befehl an Datenbank	`mym('SQL Befehl')`

6 Zusammenfassung

Die erste Aufgabe der Diplomarbeit bestand in der Generierung der Messsignale anhand physikalischer Gesetzte in SI Einheiten. Es können nicht nur komplett neue Messaufzeichnungen generiert werden, sondern auch Datensätze des aktuellen Bordcomputers Ro-Bert ausgewertet werden. Die Generierung der Signale erfolgt über Matlab m-Files. Dabei wurde darauf geachtet, dass jede Funktion eine genaue Beschreibung besitzt, damit eine Anpassungen bzw. Weiterentwicklung so unproblematisch wie möglich ist. Die erfolgreiche Bearbeitung dieser Thematik bildet die Grundlage für die beiden weiteren Punkte der Aufgabenstellung.
Die grafische Auswertung der erzeugten Messsignale kann komfortabel über ein GUI gesteuert werden. Dies ermöglicht auch unerfahrenen Benutzern schnell die gewünschten Daten zu ploten. Die Diagramme besitzen genaue Zeitangaben des Flugzeugeinsatzes und eine Datentypbeschreibung der darstellenden Größen. Desweitern werden einige Messsignale wie z.B. Drehung des Flugzeuges und zeitlicher Verlauf einer Hagelzelle in einer Simulation visualisiert, da so der Sachverhalt für den Benutzer viel anschaulicher ist als nur in einem Diagramm.
Als letze Aufgabe dieser Diplomarbeit wurde eine Möglichkeit geschaffen, die in Matlab generierten Messsignale auf einen MySQL Server zu übertragen. Hierbei ging es primär darum zu prüfen, wie leistungsstark eine derartige Verbindung ist. In den durchgeführten Testläufen gab es keinerlei Probleme, auch wenn große Datenmengen übertragen wurden. Weiterer Pluspunkt der in der Diplomarbeit erarbeiteten Schnittstelle ist, dass sie kostenfrei ist und somit keine Abhängigkeit einer Matlab Toolbox besteht.
Durch die Verschmelzung der einzelnen Programmteile wurde ein Softwaretool auf der Basis von Matlab geschaffen welches es ermöglicht, vollautomatisch Flugeinsätze grafisch auszuwerten und die daraus gewonnenen Daten in eine MySQL Datenbank zu speichern. Durch das Setzen von Parametern ist es weiterhin möglich, alle Messsignale, die der zukünftige Bordrechner HAIL misst, ebenfalls vollautomatisch zu erzeugen und anschließend grafisch auszuwerten.

7 Ausblick

Die Berechnungen, die für die Generierung der Messsignale nötig sind, werden über Matlab Funktionen ausgeführt. Ebenfalls ist das Matlab - MySQL Interface über Funktionen realisiert. Ein erweiternder Schritt wäre nun, diese Berechnung in einem Simulink Modell durchzuführen. Ein großer Vorteil ist die leistungsstarke 3D Visualisierung via Simulink 3D Animation, da der Standard Plot Befehl aufgrund der großen Datenmenge jetzt schon bis an die Leistungsgrenze belastet wird. Desweitern könnte über ein Simulink Modell die Echtzeitübertragung der MySQL Verbindung geprüft und der zukünftige Bordrechner HAIL in einer Hardware-in-the-Loop Simulation auf Herz und Nieren getestet werden. Vorteil wäre ganz klar, dass alles auf der Basis von Matlab & Simulink durchgeführt würde und deshalb keine Kompatibilitätsprobleme mit anderen Programmen bestünden.
Mit mehr Kartenmaterial des Bayrischen Vermessungsamtes könnte die komplette Topologie des Einsatzgebietes detaillierter dargestellt werden. Vorgehensweise wäre, die zweidimensionale Landkarte über die dreidimensionale Topologie als Textur zu legen. Mit Testdaten des Stadtbereichs Rosenheim wurde es bereits erfolgreich ausgeführt.

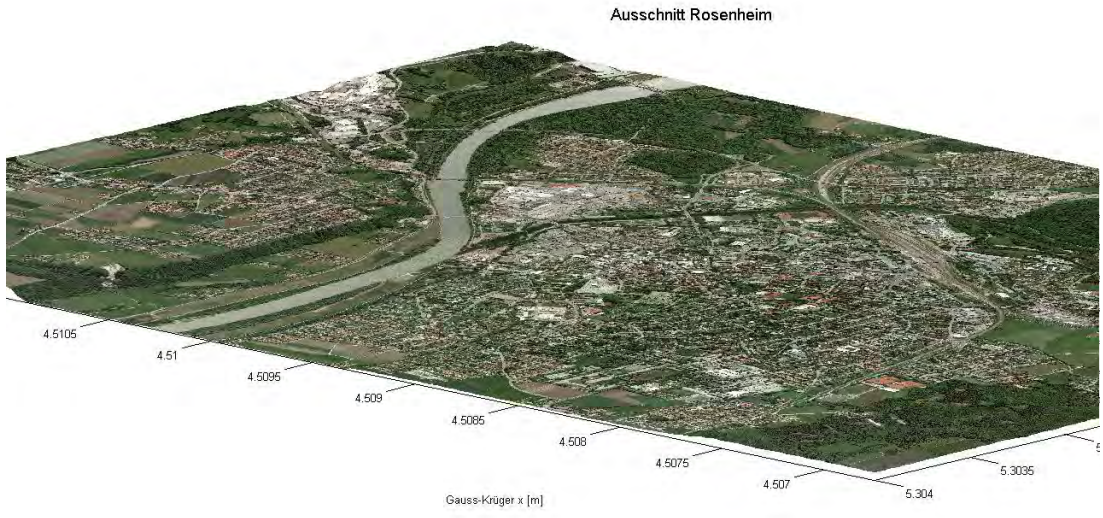


Abb. 7.1: Stadtbereich Rosenheim
Quelle: (c) Bayrisches Vermessungsamt

Wie schon in dem Programm von Tobias Hoeglauer, ist es auch in diesem möglich, Radardaten des DWD mit einzubinden. Würde der DWD diese Daten zur Verfügung stellen, könnten sich die Auswirkungen der Hagelabwehr deutlich besser bewerten lassen. Als letzter Punkt sei noch erwähnt, dass es für einige Daten, wie z.B. die elektrische Feldstärke, noch keine Referenzmessungen gibt. Die Berechnung erfolgt nach einer Modellannahme. Sobald der neue Bordrechner HAIL in Betrieb genommen wird, ist es sehr zu empfehlen, die generierten Messdaten mit einer echten Messung zu vergleichen. Falls enorme Abweichungen vorhanden sind, könnten die Berechnungsalgorithmen angepasst werden.

A Handbuch

Hochschule **Rosenheim**
University of Applied Sciences

Studiengang Produktionstechnik

DIPLOMARBEIT

Modellieren von idealen Messsignalen in Matlab/Simulink zur Verifikation von Navigationsalgorithmen zur Hagelbekämpfung

Vorgelegt von:	Maximilian Leinenbach
Matrikelnummer:	634939
am:	22.12.2011

Handbuch

Erstprüfer: Prof. Dr.-Ing. Peter Zentgraf, M.SC.

Zweitprüfer: Prof. Dr.-Ing. Franz Plötz

Vorwort

Das Matlabprogramm “IdealerSensor” wurde mit der MATLAB Version 7.10 (R2010a) auf dem Betriebssystem Windows 7 Professional 32 Bit entwickelt und getestet. Als Hardware diente ein DELL Inspiron 1520 Laptop mit 2.2 GHz, 2.0 GB Arbeitsspeicher und NVIDIA GeForce 8600M GT Grafikkarte. Die vollständige Funktionsfähigkeit kann nicht auf jedem Computer gewährleistet werden.
Die Aufgabe des Programmes “IdealerSensor” besteht darin, Messsignale anhand physikalischer Gesetze in SI Einheiten zu generieren. Es werden flugmechanische, meteorologische und geophysikalische Signale erzeugt. Das Generieren der Datensätze soll auf drei verschiedene Arten ermöglicht werden:
Ablauf_A - Bestehende Flugdaten des alten bzw. neuen Bordrechners werden ausgewertet.
Ablauf_B - Berechnungsparameter werden über Popup Fenster von dem User eingegeben.
Ablauf_C - Berechnungsparameter werden vor Programmstart in einer Konfigurationsdatei definiert.
Diese Anleitung beschreibt die Installation und Handhabung des Programms “IdealerSensor”. Es werden die verschiedenen Einstellungsmöglichkeiten erläutert, jedoch nicht die mathematischen Grundlagen der Berechnungen, diese sind ausschließlich in der Diplomarbeit enthalten.
Das MATLAB - MySQL Interface ist ein wichtiger Programmteil. Zukünftige mess- und regelungstechnische Praktikas, aber auch aktuelle Projekte der Hochschule Rosenheim werden diesen benützen bzw. in veränderter Form verwenden. Um die Anpassung des Quelltextes auch ohne der Diplomarbeit zu ermöglichen, wird die Erklärung auch in diesem Handbuch eingefügt.

A.1 Installation

A.1.1 MATLAB

Das Programm setzt eine lauffähige Version von MATLAB auf dem Computer voraus. Das Einbinden der M-Files des Programms “IdealerSensor” geschieht über, File\Set Path\Add with Subfolders. . . .

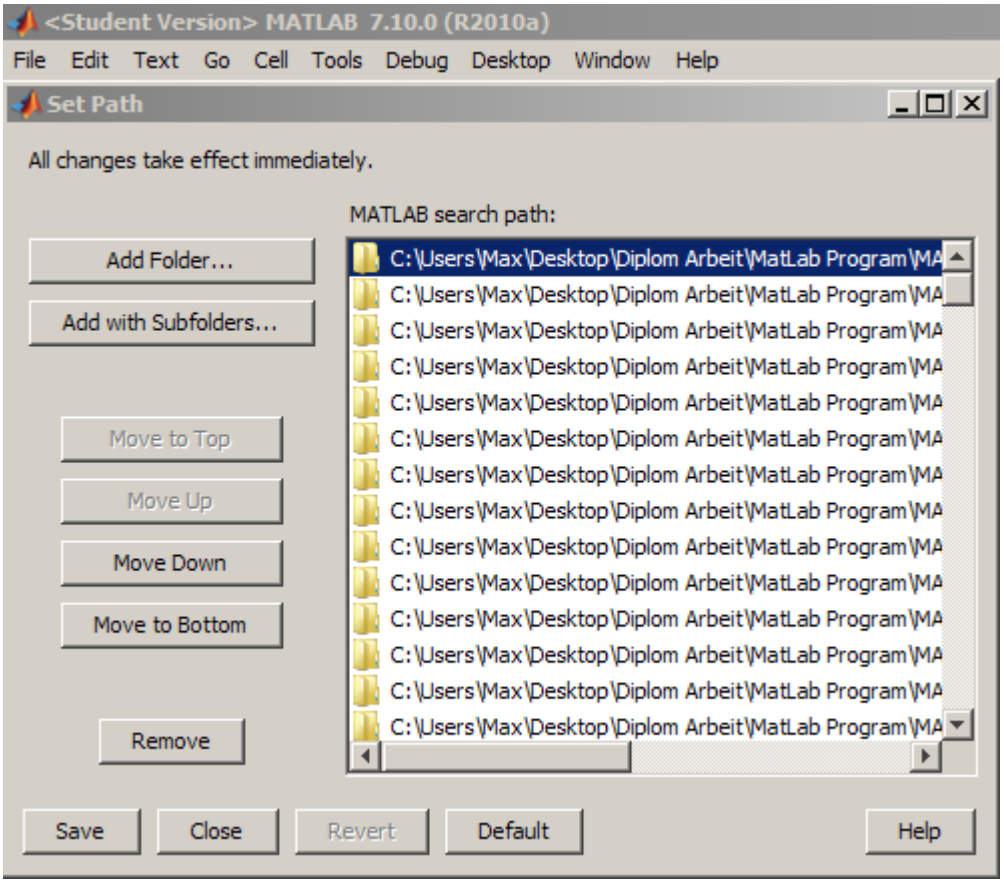


Abb. A.1: Set Path

Als Pfad wählt man *:\RO-BERTA\MATLAB\IdealerSensor\Leinenbach, falls Zugang zum Server des Mess-und-Regelungstechnik Labor[1] besteht. Andernfalls kopiert man den Ordner *:\Diplomarbeits DVD\Leinenbach von der beiliegenden DVD in einen beliebigen Ordner der Festplatte und wählt diesen Pfad.
Um Darstellungsfehler in Windows Vista und Windows 7 zu vermeiden wird empfohlen, die Aero Effekte auszuschalten. Rechtsklick auf den Desktop\Anpassen\Fensterfarbe - Windows Klassisch.

[1] ZENTGRAF_MESS-UND-REGELUNGSTECHNIK (\\TROUBADIX\PUBLIC1\LABORATORY)

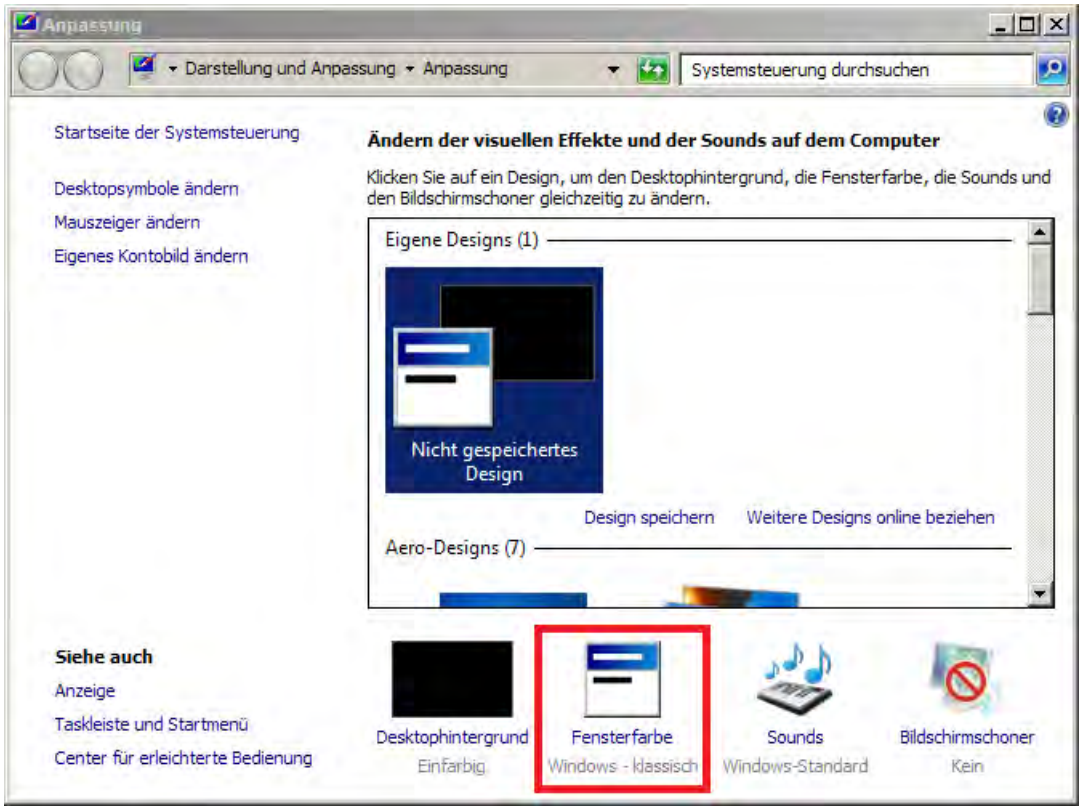


Abb. A.2: Aero Effekte abschalten

A.1.2 MySQL

Zur Installation des mYm-Interface muss die Datei mym_bin_v1.0.8.msi gestartet werden. Sie befindet sich auf *:\RO-BERTA\MATLAB\IdealerSensor\MySQL - Matlab\mym und *:\Diplomarbeit CD\MySQL - Matlab\mym.

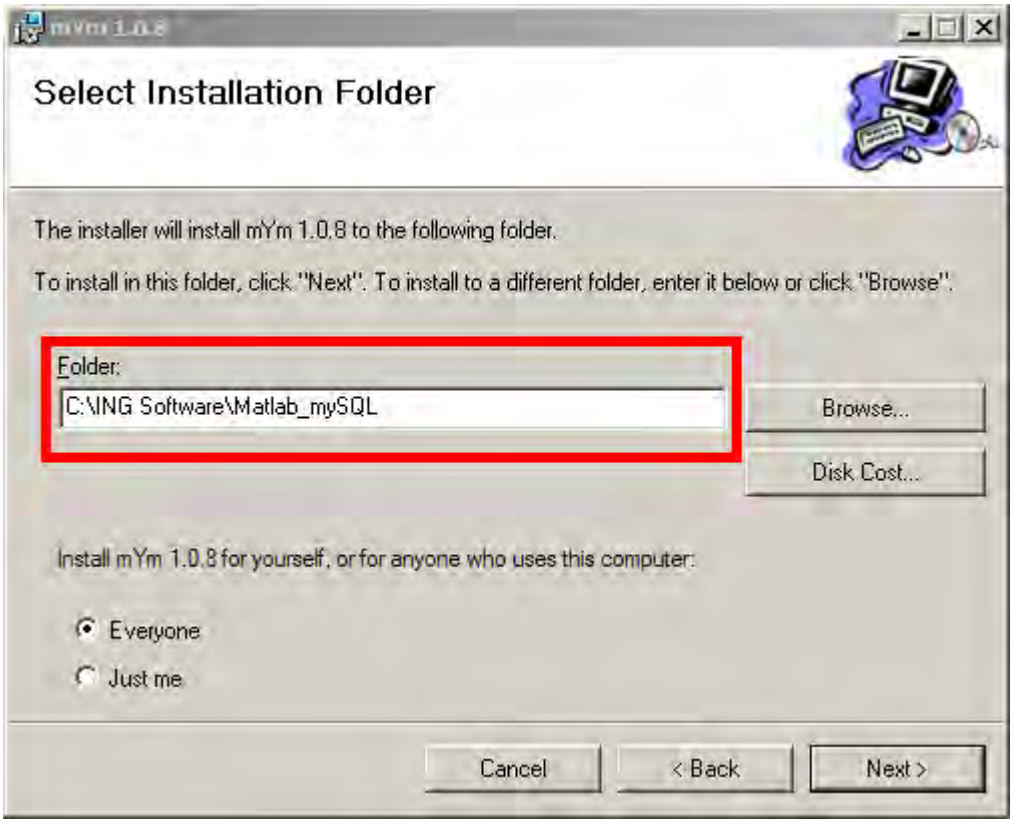


Abb. A.3: Installation mYm

Der variable Installationspfad muss auch wieder in MATLAB eingebunden werden. (A.1.1)
Um eine Verbindung mit dem Server des Mess-und-Regelungstechnik Labor herstellen zu können, ist desweiteren noch eine VPN Verbindung nötig.

A.2 Bedienungsanleitung

A.2.1 Ablauf A

A.2.1.1 Ablauf und Auswertung

Zur Auswertung eines Flugeinsatzes wird der Befehl `Ablauf_A` gestartet.
Es kann ein Ro-Bert Datensatz ausgewählt werden, dabei ist darauf zu achten, dass der Dateiname "HAGEL DDMMYY" bzw. "HAGEL DDMMYYYY" eingehalten wird.

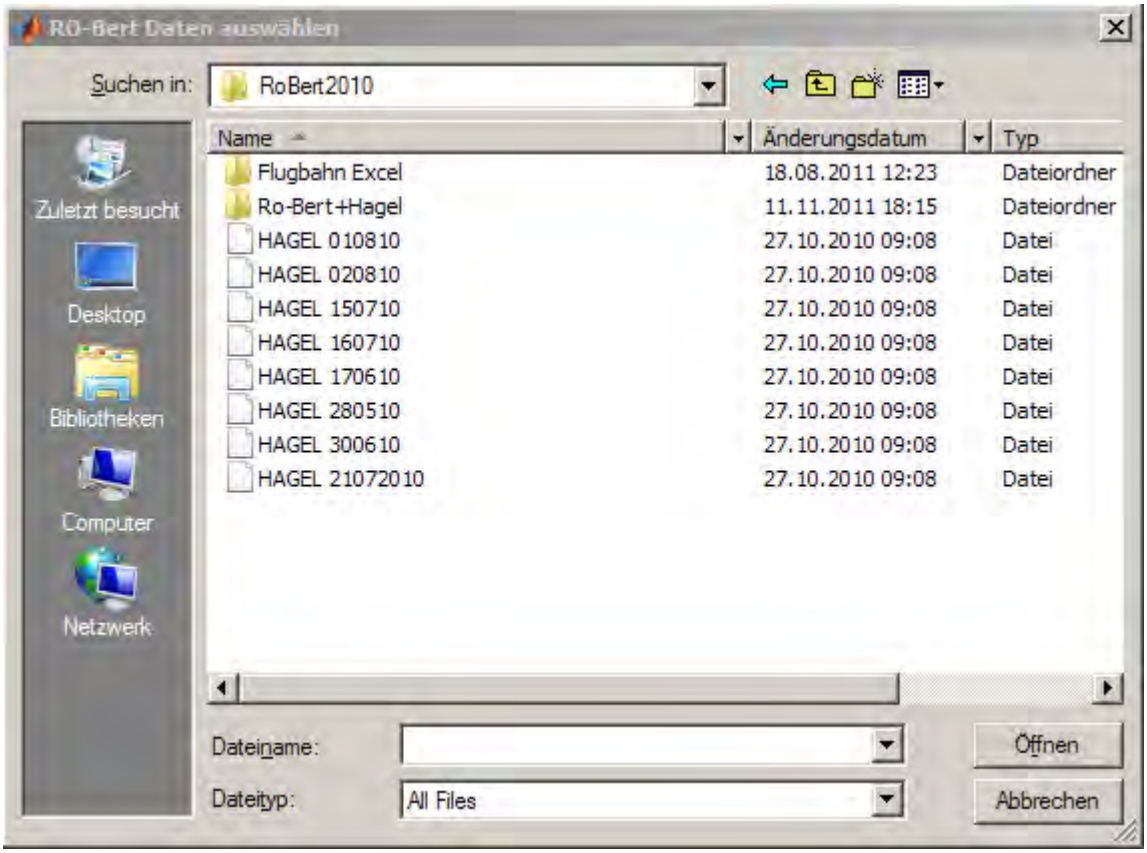


Abb. A.4: Auswahl des RO-Bert Datensatz

Nach der Berechnung wird die grafische Auswertung gestartet. Diese kann der Benutzer mit Klick auf “Fertig” beenden. Die Berechnungsergebnisse liegen im Workspace für weitere Bearbeitungen bereit.

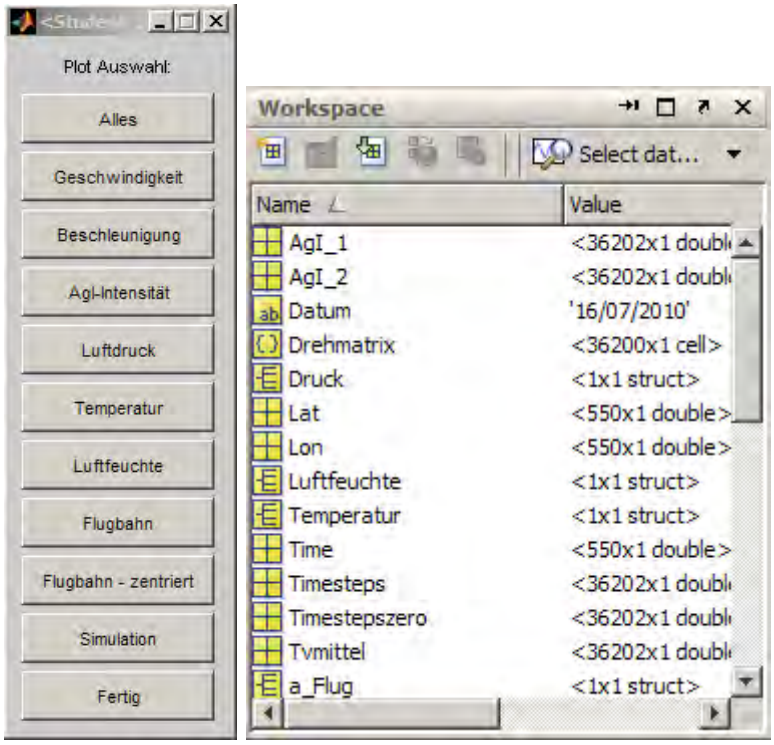


Abb. A.5: Auswertung Ablauf_A und Workspacevariablen

A.2.1.2 MySQL

Mit dem Befehl `MySQL_write_A` kann der Workspace auf den MySQL Server übertragen werden. Die Variablen werden in der Datenbank IdealerSensor/ROBERT/ mit dem Datum als Tabellenname gespeichert.
Mit dem Befehl `MySQL_read_A` werden Messdaten aus der Datenbank in den Workspace geladen und fehlende Messgrößen generiert, z.B. die 3x3 Drehmatrix. Eine Auswertung wie bei neu erzeugten Datensätzen ist somit möglich.

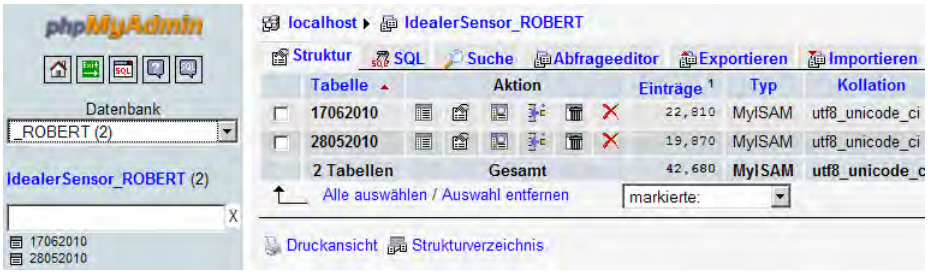


Abb. A.6: MySQL Server

A.2.2 Ablauf B

Das Programm wird mit dem Befehl `Ablauf_B` gestartet.
Bei der Eingabe der Variablen, ist das vorgegebene Format zu verwenden. Der Benutzer hat die Möglichkeiten einige Messsignale zu beeinflussen, andere Größen wie Geschwindigkeit, Beschleunigung, etc. werden vollautomatisch generiert.

A.2.2.1 Startwerte

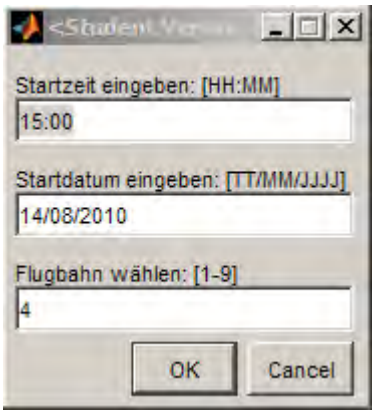


Abb. A.7: Eingabe: Startwerte

A.2.2.2 Beschleunigungssensoren

Falls virtuelle Beschleunigungssensoren berechnet werden sollen, muss der Abstand zum Flugzeugneutralpunkt in x-,y- und z-Achse angegeben werden. Es können beliebig viele Sensoren eingegeben werden.
Beispielsweise falls nur 1 Sensor berechnet werden soll, muss bei der Eingabe Beschleunigungssensor: 2 “Fertig” gewählt werden.

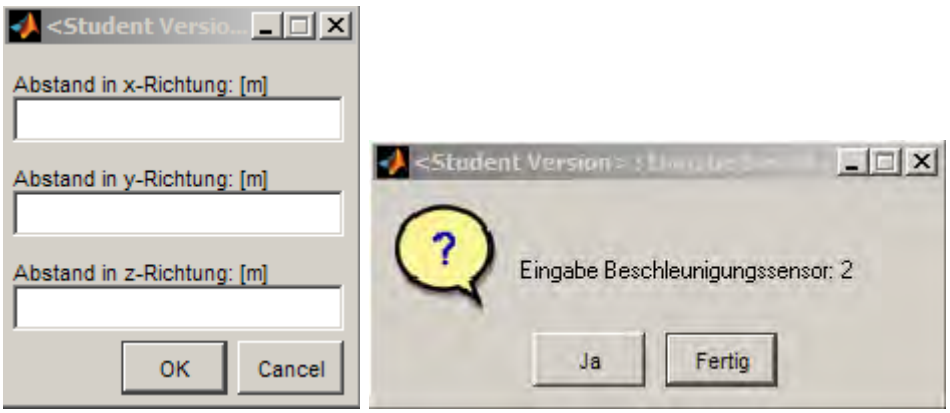


Abb. A.8: Eingabe: Beschleunigungssensoren

A.2.2.3 Wetterlage

Durch die Eingabe der Wetterlage wird die Berechung der Luftfeuchte beeinflusst. Bei wolkenlosem Himmel nimmt die Luftfeuchte mit der Höhe ab, bei wolkenbedecktem Himmel kann die Luftfeuchte jedoch auch mit der Höhe zunehmen .

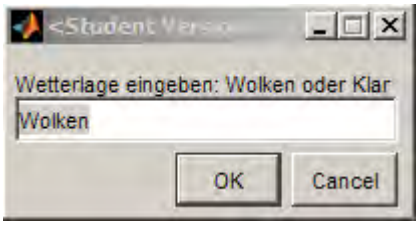


Abb. A.9: Eingabe: Wetterlage

A.2.2.4 Wind

Die Zugrichtung der Wolke, aber auch der dynamische Druck, wird durch die Windrichtung und Windstärke definiert. Die Windrichtung ist die Richtung, aus der der Wind weht. Die Dauer definiert den Zeitbereich, in dem der Wind die eingegeben Eigenschaften besitzt. Die Eingabe geschieht solange, bis der komplette Flugzeitraum definiert ist. Durch die mehrfache Eingabe ist es möglich, Windgeschwindigkeits- und Windrichtungswechsel zu simulieren.

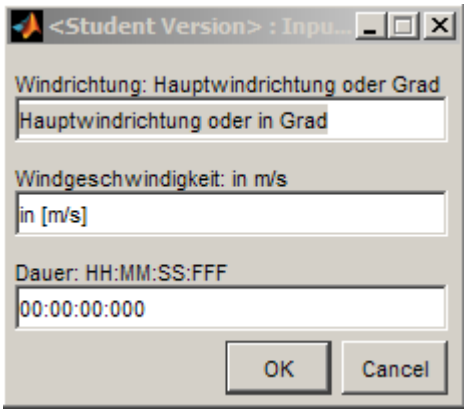


Abb. A.10: Eingabe: Wind

A.2.2.5 AgI Klasse

Das Hagelflugzeug verfügt über zwei Silberjodidkanonen, welche unabhängig voneinander betrieben werden. Die Menge des ausgestoßenen Silberjodids wird über eine digitale Steuerung bestimmt. Durch Eingabe der Intensitätsklasse oder eines numerischen Bit Wertes kann diese simuliert werden. Die Definition des Zeitbereiches geschieht analog zur Eingabe der Windparameter. A.2.2.4

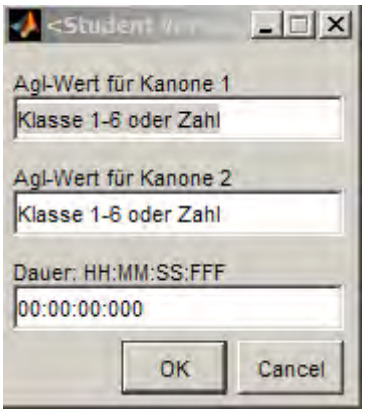


Abb. A.11: Eingabe: Silberjodid

A.2.2.6 Auswertung

Nach der Berechnung wird die grafische Auswertung gestartet, diese kann der Benutzer mit Klick auf “Fertig” beenden. Die Berechnungsergebniss liegen im Workspace für weitere Bearbeitungen bereit.

A.2.3 Ablauf C

Vor dem Programmstart muss eine Konfigurationsdatei in den Workspace geladen werden. Diese definiert alle Programmparameter, änhlich der Eingabeaufforderungen des Programmablaufes B. Eine vordefinierte Konfigurationsdatei wird über `load Konfiguration_Standart` geladen. Nachdem alle Einstellungen in der Konfiguration angepasst wurden, kann das Programm mit dem Befehl `Ablauf_C` gestartet werden.

A.2.3.1 Konfigurationsdatei

Die Strukturvariablen können entweder über die Kommandozeile, oder über den Workspace angepasst werden . Auch bei dem Ablauf C ist darauf zu achten, dass das vorgebebene Variablenformat beibehalten wird.
Beispiel:
Startzeit und Datum definieren den Startzeitpunkt am Flughafen Vogtareuth.

```
% Lade vordefinierte Konfiguration
load Konfiguration_Standart
% Ändern des Datums
Konfiguration.Datum='25/11/2012';
% Ändern der Startzeit
Konfiguration.Startzeit='10:40';
```

Abb. A.12: Struktur der Konfigurationdatei

A.2.3.2 Flugbahn

Es besteht die Möglichkeit zwischen neun Flugbahnen zu wählen, Flugbahn_1 bis Flugbahn_9. Leider ist in der ersten Version des Programms “IdealerSensor” nur Oberbayern als 3D Topologie vorhanden. Flugbahn_4 ist die einzige, bei der die gesamte Flugstrecke über bayrischem Hoheitsgebiet verläuft und deshalb voreingestellt. Es können andere Flugbahnen benützt werden, jedoch kommt es bei Berechnung einiger Messsignale zu unrealistischen Ergebnissen.

A.2.3.3 Wetter

Einstellungsmöglichkeiten werden in A.2.2.3 beschrieben.

A.2.3.4 Sensoren

Die Positionen der virtuellen Beschleunigungssensoren werden über einen Matrix definiert. Dabei sind die x-,y- und z-Werte der jeweilige Abstand zum Flugzeugneutralpunkt. Es besteht die Möglichkeit, bis zu n Sensoren zu berechnen.

$$Sensoren = \begin{Bmatrix} x_1 & ... & x_n \\ y_1 & ... & y_n \\ z_1 & ... & z_n \end{Bmatrix}$$

A.2.3.5 Wind

Die Windberechnung wird über die Zeilenvektoren: Richtung, Geschwindigkeit und Dauer definiert. Diese drei Vektoren haben die gleiche Länge $nx1$. Mit der Variablen "Methode" wird die Berechnungsart gewählt, diese kann entweder Normal oder mit Interpolation ablaufen.

A.2.3.6 Richtung

Die Eingabe der Windrichtung erfolgt in Grad, dabei entspricht $0^{\circ} = Norden$, $90^{\circ} = Osten$, $180^{\circ} = Süden$ und $270 = Westen$.
Beispiel: Abbildung A.13

$$Richtung = \begin{Bmatrix} 45 \\ 250 \\ 184 \end{Bmatrix}$$

A.2.3.7 Geschwindigkeit

Die Eingabe der Geschwindigkeit erfolgt in $\frac{m}{s}$.

Beispiel: Abbildung A.13

$$Geschwindigkeit = \begin{Bmatrix} 14 \\ 26 \\ 20 \end{Bmatrix}$$

A.2.3.8 Dauer

Die Eingabe des Zeitformates erfolgt in HH:MM:SS:FFF, dabei ist darauf zu achten das die Zeiten aufsummiert werden.
Beispiel: $\Delta Zeit = 15min$ Abbildung A.13

$$Dauer = \begin{Bmatrix} 00:15:00:000 \\ 00:30:00:000 \\ 00:45:00:000 \end{Bmatrix}$$

Desweitern muss der komplette Flugzeitbereich definiert werden. Falls das letzte Vektorelement die eigentliche länge der Flugzeit überschreitet, wird automatisch der Zeitpunkt $Flug_{Ende}$ verwendet.

A.2.3.9 Methode

Wird die Variable Methode von `'default'` auf `'interp'` gesetzt, werden die Werte zwischen den vorgegebenen Zeitpunkten interpoliert.

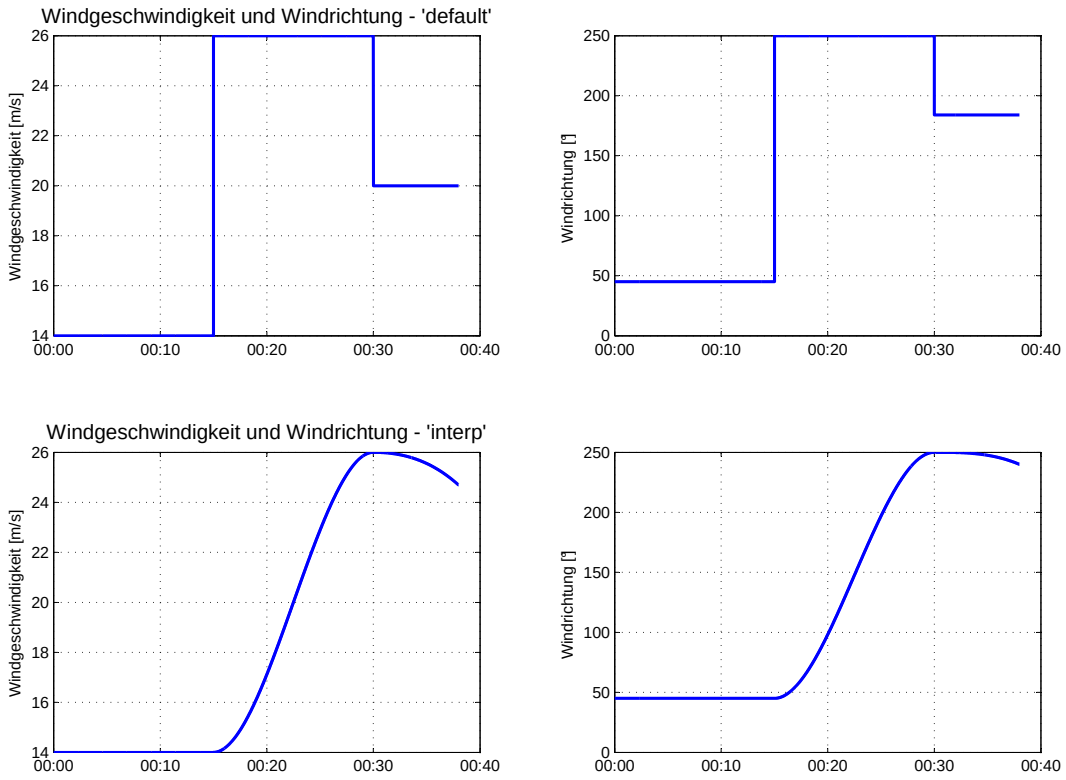


Abb. A.13: Methodenvergleich Wind

A.2.3.10 Agl Kanonen

Die Konfiguration der Berechnung erfolgt analog zu der vorhin erklärten Windberechnung. Statt Windrichtung und Windstärke müssen lediglich die Zeilenvektoren Kanone_1 und Kanone_2 mit den simulierten Bit - Werten definiert werden.

A.2.3.11 Interpolation

Die Berechnung der Feinstaubdichte, Lufttemperatur und Luftfeuchte wird mit Hilfe vorhandener Messdaten des Jahres 2009 durchgeführt. Die Positionen und Zeitpunkte der Messungen der Stationen sind bekannt, dadurch kann in einem dreidimensionalen Raum ein zeit- und ortsabhängiger Wert interpoliert werden. Die Zeitrasterung der berechneten Messsignale beträgt $0.1s$. Da eine Interpolation für jeden Zeitwert sehr rechenintensiv ist, wurde hier die Möglichkeit geschaffen mit `'schnell'` die Messsignale nur noch ortsabhängig zu interpolieren.

$$Zeitpunkt_{1:end} = Zeitpunkt_1$$

Mit `'genau'` erfolgt eine zeit- und ortsabhängige Interpolation.
In Abbildung A.14 werden beide Methoden verglichen. Zweckmäßigerweise sollte bei Computern mit geringer Rechenleistung die Methode `'schnell'` gewählt werden.

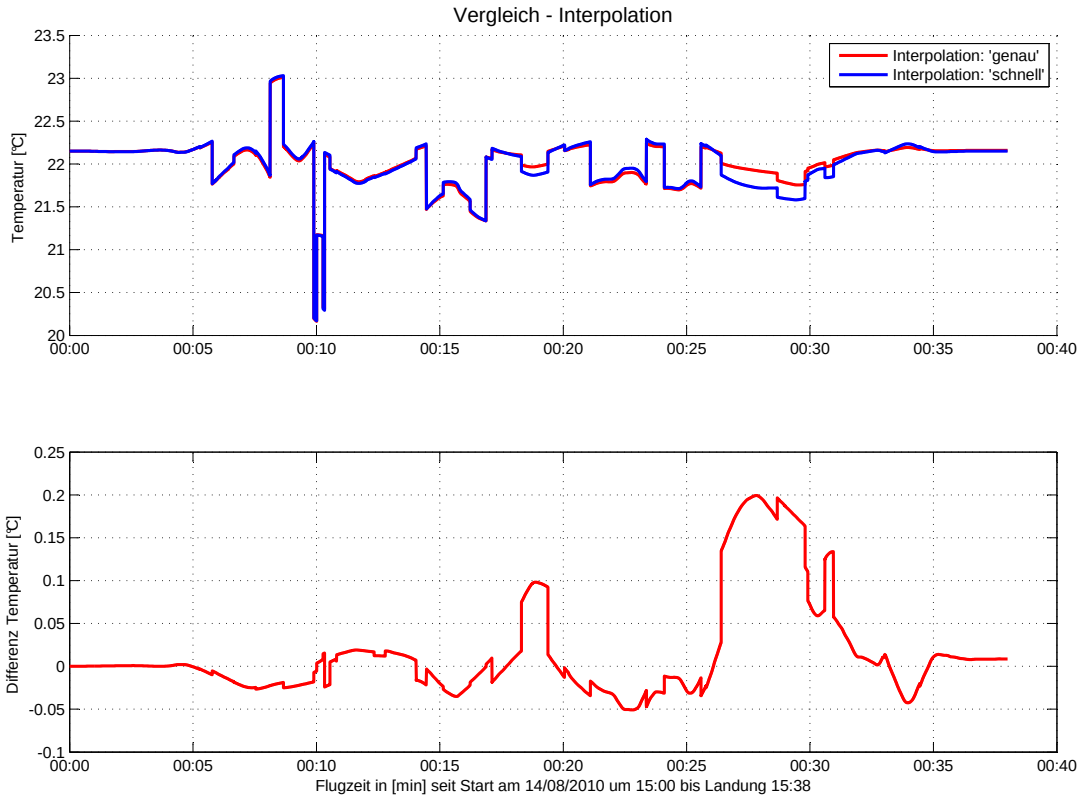


Abb. A.14: Vergleich Interpolationsmethoden

A.2.3.12 Plotbereich

Wird die Variable mit `'Alles'` belegt, wird der komplette Messbereich dargestellt.
Wird die Variable mit `'Abschnitt'` belegt, wird der Messbereich am Anfang und Ende um ca. 5 Minuten abgeschnitten. Dies verhindert, dass die automatische Achsenskalierung von MATLAB durch zu extreme Werte weiteres Auswerten der Plots unmöglich macht.

A.2.3.13 Hagelwolke

Mit dieser Variable kann lediglich die Form der simulierten Hagelwolke geändert werden. Standartmäßig gibt es eine vordefinierte Form, `'Wolke_1'`.

A.3 Matlab - MySQL

Die Verbindung zwischen MATLAB und MySQL wird durch einen in MATLAB kompilierten C-Code hergestellt. Das mYm-Interface[2] ermöglicht Befehle an eine Datenbank in normaler SQL Syntax zu senden. Um die Kommunikation zwischen MATLAB und Datenbank zu erleichtern, werden zudem die mYm-Utilities[3] verwendet. Bei den mYm-Utilities handelt es sich um Functions, die Befehle in SQL Syntax übersetzen. Der schematische Verbindungsablauf ist in Abbildung 3.1 dargestellt.

[2] mYm v1.0.8, Copyright (C) 2006, Swiss Federal Institute of technology, Lausanne, CH
[3] mYm Utilities, 2007, Dimitri Shvorob

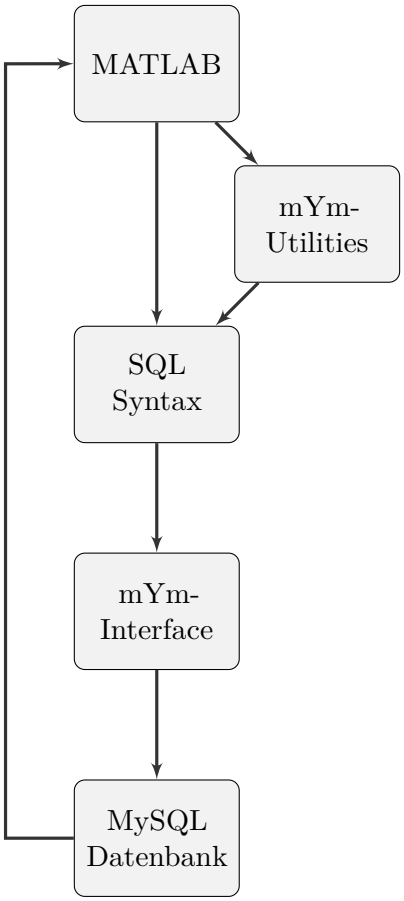


Abb. A.15: Kommunikation MATLAB-MySQL

A.3.1 mYm-Utilities

Bei der Bedienung der mYm-Utilities sind einige grundsätzliche Regeln zu beachten. Es können nur Daten des Datentyps `char` in eine Datenbank übermittelt werden. Zahlenformate müssen folglich in einen Character-String umgewandelt werden. Die übermittelten Zahlen des Datentypes `char` werden automatisch in das definierte Spaltenformat der Datenbank konvertiert, z.B. `double`. Ausserdem sind Inputvariablen, Outputvariablen, Spaltennamen, und Spaltendatentypen in einem jeweils seperaten Cell-Array zu definieren.

A.3.1.1 Beispiel

Anhand eines Beispiels soll der Mechanismus genauer erläutert werden:
`Leinenbach\MySQL\Beispielprogramm.m`

```
%Beispielprogramm MATLAB–MySQL
%
%write
%
```

```
%Server Me-Re-Technick
host='141.60.140.43';
user='leinenbach';
pwd='leinenbach';
%Herstellen Verbindung
myopen(host,user,pwd);
%Prüft die Verbindung
con=myisopen;
if con==1
disp(['Verbindung zum Server: ', host, ' hergestellt']);
else
disp(['Verbindung konnte nicht hergestellt werden']);
end
%Anzeigen allder Datenbanken
dblist
%Öffnet D.B.
dbopen('IdealerSensor_Beispiel')
%Anzeigen aller Tabellen
tblist
%Array der Spaltennamen
Spaltennamen={'Nachname', 'Vorname', 'Geburtsjahr'};
%Array des Datenformates
Datentyp={'varchar(30)', 'varchar(30)', 'double'};
%Erstellen einer Tabelle
%Replace, damit Beispiel reproduzierbar ist
tbadd('Mitarbeiter',Spaltennamen,Datentyp,'replace');
%Inputvariablen
Nachname_sql={'Müller', 'Meier'};
Vorname_sql={'Sepp', 'Hans'};
Geburtsjahr_sql={'1982', '1945'};
%Übergabearray Definieren
Inputvariablen={'Nachname_sql', 'Vorname_sql', 'Geburtsjahr_sql'};
%Datenübertragung Matlab-MySQL
tbwrite('Mitarbeiter',Spaltennamen,Inputvariablen);
%
%read
%
%Verbindungsaufbau analog zu write
%Workspacevariablen definieren
Outputvariablen={'Nachname_read', 'Vorname_read', 'Geburtsjahr_read'};
%Datenübertragung MySQL-Matlab
%'' = Art der Datenauswahl -> Alle
tbread('Mitarbeiter',Spaltennamen,Outputvariablen,'');
%Verbindung Beenden
myclose;
```

SQL-Abfrageergebnis

Host: localhost
Datenbank: IdealerSensor_Beispiel
Erstellungszeit: 25. November 2011 um 14:42
Erstellt von: phpMyAdmin 3.3.10 / MySQL 5.1.46-log
SQL-Befehl: SELECT * FROM `Mitarbeiter` LIMIT 0, 30 ;
Zeilen: 2

Nachname	Vorname	Geburtsjahr
Müller	Sepp	1982
Meier	Hans	1945

Abb. A.16: Datenübertragung Beispielprogramm.m

A.3.1.2 Funktionstabelle

Übersicht aller verfügbaren mYm-Utilities Funktionen.

Tab. A.1: mYm-Funktionen

Funktion:	Aufgabe:	Syntax:
myisopen	Prüfen einer MySQL Verbindung	`myisopen`
myopen	Herstellen einer MySQL Verbindung	`myopen('Host','User','Passwort');`
myclose	Schließen einer MySQL Verbindung	`myclose`
dblist	Auflisten aller Datenbanken	`dblist`
dbopen	Öffnet Datenbank	`dbopen('Datenbank');`
dbcurr	Zeigt aktuelle Datenbank an	`dbcurr`
dbadd	Erstellen einer Datenbank	`dbadd('Datenbank');`
dbdrop	Löschen einer Datenbank	`dbdrop('Datenbank');`
tblist	Auflisten aller Tabellen	`tblist`
tbadd	Erstellen einer Tabelle	`tbadd('Tabelle',{'Spalten'},...` `{'Datentypen'};`
tbdrop	Löschen einer Tabelle	`tbdrop('Tabelle');`
tbrename	Umbenennen einer Tabelle	`tbrename('Tabelle',...` `'Tabelleneu');`
tbattr	Auflisten Spaltennamen und -typen.	`[Namen, Typen]=tbattr('Tabelle')`
tbsize	Tabellendimension Zeilen x Spalten	`[size]=tbsize('Tabelle');`
tbread	Lesen aus Tabelle	`tbread('Tabelle',{'Spalten'},...` `{'Outputvariablen'});`
tbwrite	Schreiben in Tabelle	`tbwrite('Tabelle',{'Spalten'},...` `{'Inputvariablen'});`
mym	Sendet SQL Befehl an Datenbank	`mym('SQL Befehl')`

Literaturverzeichnis

[BSMM08] Bronstein ; Semendjajew ; Musiol ; Muehling: *Taschenbuch der Mathematik.* 7. Auflage. Verlag Harri Deutsch, 2008

[GF04] Grote ; Feldhusen: *Dubbel Taschenbuch fuer den Maschinenbau.* 21. Auflage. Springer, 2004

[Hoe09] Hoeglaueri, Tobias: *Diplomarbeit Aufbereitung und Online-Visualisierung der Messdaten eines Hagelabwehrflugs.* 2009

[LC84] Liljequist, Goesta ; Cehak, Conrad: *Allgemeine Meteorologie.* 3. Auflage. Springer, 1984

[Leu04] Leute, Ulrich: *Physik und ihre Anwendungen in Technik und Umwelt.* 2. Auflage. Hanser, 2004

[Mal06] Malberg, Horst: *Meteorologie und Klimatologie: Eine Einfuehrung.* 5. Auflage. Springer, 2006

[Pie06] Pietruszka, Wolf D.: *Matlab und Simulink in der Ingenieurspraxis - Modellbildung, Berchnung und Simulation.* 2. Auflage. Teubner, 2006

[Sei06] Seideli, Henrik: *Die Mathematik der Gauss-Krueger-Abbildung.* 2006

[Sla99] Slabaugh, Gregory: Computing Euler Angles from a Rotation Matrix. (1999)

[Tew07] Tewari, Ashish: *Atmospheric and SpaceFlight Dynamics.* 1. Auflage. Birkhaeuser, 2007

[Zen11] Zentgraf, Peter: Skript. (2011)

Abbildungsverzeichnis

Tabellenverzeichnis