Simulation des Verhaltens einer low-cost Strapdown *IMU* unter Laborbedingungen

RAUL DOROBANTU [*])

Abstract

The paper presents some results of a labor simulation for a low cost *IMU* (Inertial Measurement Unit) with strapdown mechanisation. In the present approach we have neglected Earth rotation and gravity variations, because of the poor gyroscope sensitivities of our low-cost *ISA* (Inertial Sensor Assembly) and because of the small area of the trajectory.
The aim of this experiment was to test the implementation of the principal steps of a strapdown navigational algorithm, the modelling and calibration of the *IMU*, as well as the evaluation of the influence of diverse implementation parameters, as the integration step and method.
After a short presentation of the principal inertial navigation systems and of the used *ISA* sensors, we are focused on the implementation in an interactive manner of a simple strapdown mechanisation of a low-cost *ISA* type *Tgac – iMAR*.
This was realised by means of the graphical programming language *Simulink* (*Matlab*), with the aim to construct the trajectory – as well as the intermediate signal visualisation. In a first step one has modelled only the bias, scale-factor, non-orthogonalities and the linear drift of the inertial sensors; the implementation of others supplementary complex modelling parameters (as temperature effects, superior terms of the polynomial drifts, etc.) can be done directly.
From the presented results we conclude that steps up to 50 Hz (to diminish the commutation errors of the rotational sequences) and integration methods superior of Runge-Kutta ord. 4 (we have tried the Euler, Adams-Gear and Runge-Kutta ord. 3-5 integration methods, too) are recommended. The *Simulink* implementation constitutes a veritable evaluation tool for the performance of an *ISA* unit, with its rapid and transparent modelling features.
For the interested reader the *Simulink* source programs, as well as some 2D simulation demo examples are available.

1. Mechanisierung von Trägheitsnavigationssystemen

Das zentrale Problem der Trägheitsnavigation ist die autonome Bestimmung der beispielsweise von einem Fahrzeug zurückgelegten Trajektorie, der Orientierung und der Navigationsparameter (Geschwindigkeiten, Beschleunigungen) aus den von der *IMU* (Inertial Measurement Unit) gelieferten Daten. Für die Bestimmung dieser Parameter ist ein Referenzsystem festzulegen.
Die drei Hauptsysteme der Trägheitsnavigation [*Heinze, 1996*] sind folgende:

I. Räumlich stabilisierte Trägheitsplattformen (siehe die vereinfachte Blockschaltung von Abb. 1.1), wobei die kardanisch aufgehängte Plattform die Orientierung in Bezug auf ein inertiales Referenzsystem beibehält (realisiert mit Hilfe einer Rückkopplungsschleife basierend auf Kreiselablesungen).

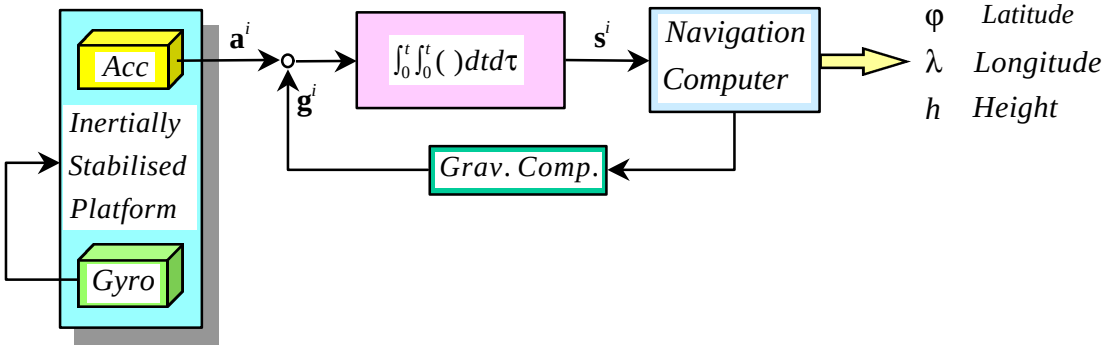


Abb. 1.1 *Trägheitsnavigationssystem mit raumstabilisierter Plattform*

Durch eine Doppelintegration der spezifischen Kraft $\mathbf{f}^i$ (im voraus korrigiert um den Einfluß der Schwerebeschleunigungsvariationen [*Rummel, 1986*]) erhält man die Trajektorienparameter im inertialen Referenzsystem. Daraus lassen sich terrestrische Koordinaten ableiten. Dieses System hat seine Vorteile in der

[*]) Technische Universität München, Institut für Astronomische und Physikalische Geodäsie, Arcisstr. 21, D-80290 München, Germany; **email: raul@alpha.fesg.tu-muenchen.de**

Einfachheit der Berechnung und in der optimalen Nutzung des Auslegungsbereichs der Trägheitssensoren, die bei diesem System voll von der Dynamik des tragenden Fahrzeugs isoliert sind; seine Nachteile bestehen in der Größe, in der anspruchsvollen Wartung des mechanischen Systems, in der geringen Zuverlässigkeit und in relativ hohen Preisen.

II. Strapdown-Systeme (siehe Blockschaltung von Abb. 1.2), die starr auf dem tragenden Fahrzeug befestigt sind, ändern ständig die Orientierung.

Für die Umwandlung der Komponenten der spezifischen Kraft $\mathbf{a}^b$ vom körperfesten Koordinatensystem (*b – body)* in das inertiale Referenzsystem (*i*), in dem man die Berechnungen ausführt, ist es nötig, für jeden Schritt die Rotationsmatrix $\mathbf{C}_b^i$ zu bestimmen. Dabei nutzt man die vom Kreisel gelieferten Winkelgeschwindigkeits-informationen durch Integrieren eines Differentialgleichungssystems der Form: $\dot{\mathbf{C}}_b^i = \mathbf{C}_b^i \cdot \Omega_{ib}^b$ [*Britting, 1971*], mit der schief-symmetrischen Matrix $\Omega_{ib}^b = \boldsymbol{\omega}_{ib}^b \times$, abgeleitet aus dem Winkelgeschwindigkeitsvektor $\boldsymbol{\omega}_{ib}^b = [\omega_{ibx}^b \quad \omega_{iby}^b \quad \omega_{ibz}^b]^T$ des mobilen Fahrzeugs im inertialen Referenzsystem. Eine rechnerische Umwandlung der Vektorkomponenten der spezifischen Kraft von Fahrzeugkoordinatensystem (*b*) in das inertiale Referenzsystem (*i*) ist wegen der permanenten Änderung der Fahrzeugorientierung notwendig. Analog der effektiven Mechanisierung im Fall der stabilisierten Trägheitsplattform könnte sie als ein spezieller Typ der "Strapdown Mechanisierung" betrachtet werden.

Die Fahrzeugorientierung, eine zur Trajektorie komplementäre Navigationsgröße, wird durch Sets von Euler–Winkeln ϕ^i, θ^i, ψ^i (Rotationswinkel um die Achsen des inertialen Referenzsystems) angegeben; zum Vermeiden von Singularitäten bei 3D-Navigationslösungen, kann man die Quaternion-Parametrisierung anwenden [Titterton et al., 1997].

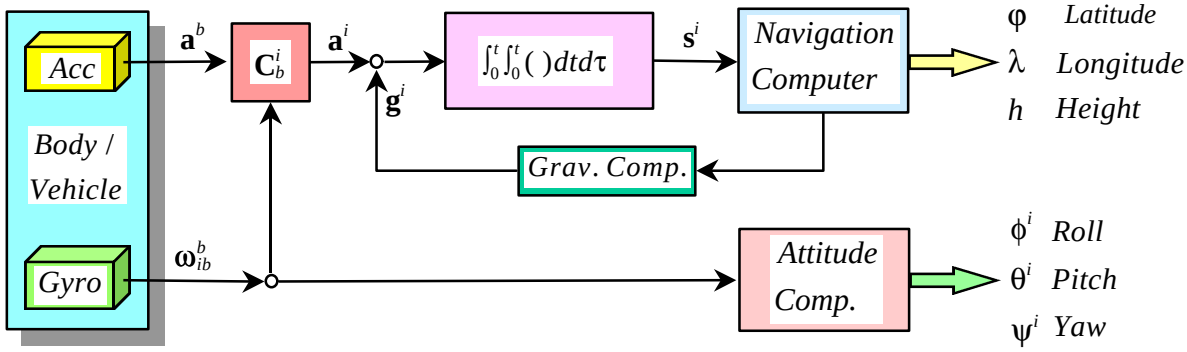


Abb. 1.2 *Strapdown Trägheitsnavigationssystem mit Berechnung im inertialen Referenzsystem*

Die wichtigsten Vorteile der Strapdown-Trägheitsnavigationssysteme bestehen in den geringen Ausmaßen, in guter Zuverlässigkeit und in relativ niedrigen Preisen sowie in der Verfügbarkeit von Navigationsinformationen (wie Beschleunigungen, Winkelgeschwindigkeiten, etc.) im Bordkoordinatensystem. Letztere werden zum Beispiel für die Autopilotfunktion benötigt.

Dabei soll auf die Schwierigkeiten in der Realisierung eines Online Systems hingewiesen werden, das permanent alle Bewegungen des Trägerfahrzeugs verfolgt: Bias- und Drift-Fehler der Trägheitssensoren müssen unter ungünstigen dynamischen Bedingungen des Inertialsensorsystems kompensiert werden.

III. Geographisch orientierte Trägheitssysteme. Am Anfang des Meßvorgangs ist die Sensorplattform horizontal und nach Norden orientiert. Danach wird die mit einem kardanischen Gehänge versehene Plattform kontinuierlich nach Norden und in die horizontale Ebene gezwungen. Die Korrektionsgröße zu dieser Orientierung ergibt das Ausgangssignal. Die „Mechanisierung" der permanenten Nordorientierung des lokalen Referenzsystems, die man auch mittels eines Strapdown-Systems rein analytisch realisieren kann (siehe Blockschaltung von Abb. 1.3), erhält man durch die Subtraktion des Vektors $\boldsymbol{\omega}_{in}^b$ (Vektor der Rotationswinkelgeschwindigkeit des lokalen Navigationssystems (*n*) bezüglich des inertialen Referenzsystems (*i*)) von dem Winkelgeschwindigkeitsvektor $\boldsymbol{\omega}_{ib}^b$. Den Vektor $\boldsymbol{\omega}_{in}^b$ erhält man aus [*Schmidt, 1978*] $\boldsymbol{\omega}_{in}^n = \left[(\dot{\lambda} + \omega_{ie}) \cdot \cos\varphi \quad -\dot{\varphi} \quad -(\dot{\lambda} + \omega_{ie}) \cdot \sin\varphi\right]^T$, wobei mit ω_{ie} der Winkelgeschwindigkeitsbetrag der

Erdrotation (*e - earth*) bezüglich des inertialen geozentrischen Referenzsystems (*i*) bezeichnet wurde. Damit erhält man den Winkelgeschwindigkeitsvektor $\boldsymbol{\omega}_{nb}^{b}$ (Rotation des mobilen Fahrzeugs (*b*) bezüglich des Navigationssystems (*n*), im Referenzsystem (*b*) ausgedrückt), der bei der Bestimmung der Rotationsmatrix $\mathbf{C}_{b}^{n}$ verwendet wird.

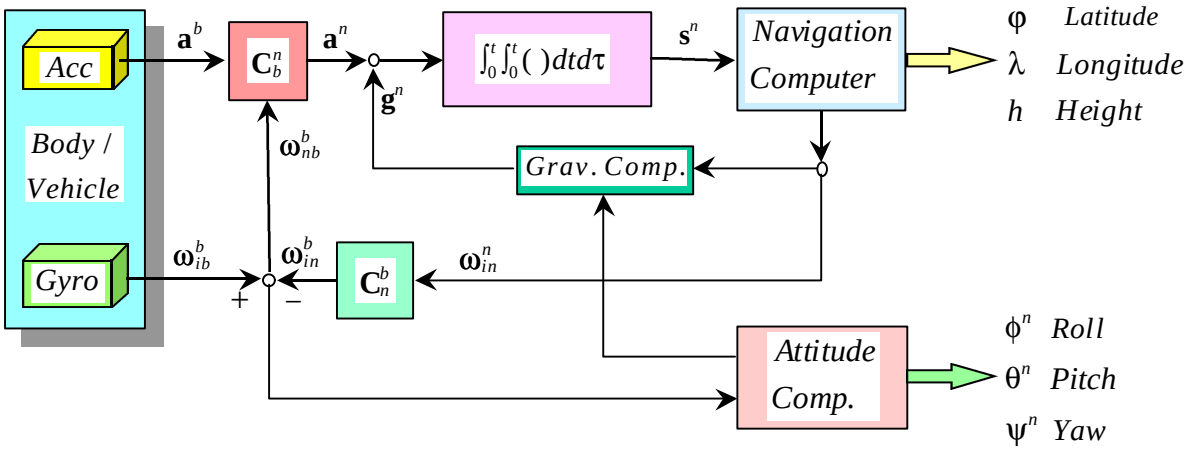


Abb. 1.3 *Strapdown Trägheitsnavigationssystem mit Berechnung im lokalen geographischen Referenzsystem*

Ein strapdown Trägheitsnavigationssystem ist wie in Abb. 1.4 strukturiert, wobei die niedrigste Ebene *ISA* aus den Trägheitssensoren besteht, die die elektrischen Signale (Größe) der Trägheitsmeßeinheit *IMU* liefern. Die *IMU* Einheit, die auf dem *ISA* vom Typ *Tgac* [*iMAR, 1995*] basiert (siehe Abb. 1.5), ist mit einem *A/D* -Wandler ausgerüstet [*National Instruments, 1995*] sowie mit einer Rechnereinheit (üblich ein *DSP* – Digital Signal Prozessor) für unter anderem Temperatur- und Fehlerausgleich (Bias, Drift).

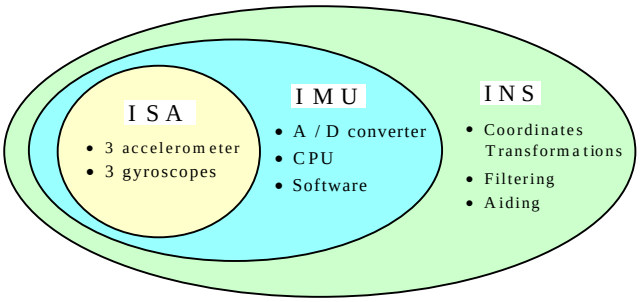


Abb. 1.4 *Struktur eines strapdown Trägheitsnavigationssystems*
ISA: Inertial System Assembly
IMU: Inertial Measurement Unit
INS: Inertial Navigation System

Das Trägheitsnavigationssystem *INS* benötigt eine zusätzliche CPU plus Software für die Komponentenumwandlung der Beschleunigungsvektoren aus dem Fahrzeugreferenzsystem in ein Inertialsystem. Nach Berücksichtigung des initialen Alignments, der Gravitations-, Coriolis- und Zentrifugalbeschleunigungskorrekturen und nach Auswertung der Daten, zusammen mit solchen aus einem unterstützenden Meßsystem (z.B. *GPS*) mit großer Zeitstabilität, in einem Kalman-Filter, erhält man üblicherweise die Navigationsdaten direkt als terrestrische Koordinaten.

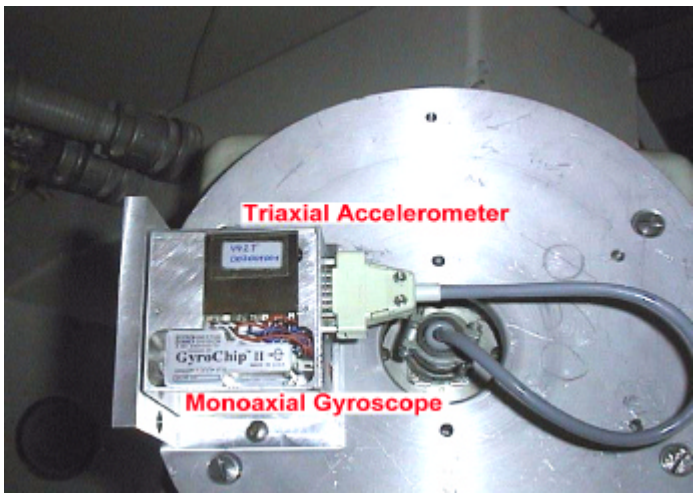


Abb. 1.5 *Das triaxiale Beschleunigungssensorensystem und ein Winkelgeschwindigkeitssensor im Inneren des Meßwürfels (ISA) – hier auf einer Drehmeßplattform plaziert*

2. Strapdown Mechanisierung einer *IMU*

Das Fehlermodell der Inertialsensoren wurde unter Verwendung von Kalibrationswerten (Bias, lineare Maßstabsfaktoren, Nichtorthogonalitäten des Kreisel-Dreibeins) aufgestellt, die vom Hersteller sowie aus zusätzlichen Kalibrationsmessungen in unserem Labor stammen. Nennenswerte Ergebnisse dieser Kalibrationsmessungen sind: die Beschreibung des Rauschverhaltens der *IMU*-Datensätze; die statische Kalibration der Beschleunigungsmesser, um zusätzlich die nichtlinearen Terme (bis zum Grad 2) der statischen Transfercharakteristik zu bestimmen; die Bestimmung des nichtlinearen Zeit- und Temperaturverhaltens der Drift und der Maßstabsfaktoren der Beschleunigungsmesser; Nichtorthogonalitäten des Beschleunigungs-messer-Dreibeins. Die komplexe Beziehung zwischen elektrischem Output-Signal u [V] eines Inertialsensors (hier: Beschleunigungsmesser) und spezifischer Kraft f, vergangener Zeit t und Temperatur T läßt sich in folgender analytischer Form darstellen [vergl. mit *Kayton, 1997*]:

$$u = D + H + k_0 + k_{01}\cdot t + k_{02}\cdot t^2 + k_1\cdot f_1 + k_2\cdot f_1^2 + k_3\cdot f_1^3 + k_{12}^{no}\cdot f_2 + k_{13}^{no}\cdot f_3 + \\ + k_{12}^{cc}\cdot f_1\cdot f_2 + k_{13}^{cc}\cdot f_1\cdot f_3 + k_{41}\cdot T + k_{42}\cdot T^2\,, \qquad (1)$$

wobei gilt

D = "toter" Bereich (unterhalb dieses Grenzwerts erhält man kein Signal),
H = Hysterese,
k_0 = Bias (je ein neuer Wert für jede neu initialisierte Messung),
k_{01}, k_{02} = Driftkoeffizienten (für das Modellieren der linearen und quadratischen Änderungen mit der Zeit),
k_1, k_2, k_3 = Polynomialkoeffizienten der nichtlinearen Antwortcharakteristik auf die spezifische Kraft entlang der Sensor-Inputachsen,
f_1, f_2, f_3 = spezifische Kraft, entlang der drei Inputachsen (bezeichnet mit 1, 2 und 3)
k_{12}^{no}, k_{13}^{no} = Kopplungskoeffizienten zwischen den Inputachsen wegen Nichtorthogonalität,
k_{12}^{cc}, k_{13}^{cc} = Kreuzkopplungskoeffizienten,
k_{41}, k_{42} = lineare und quadratische Polynomialkoeffizienten wegen der Abhängigkeit des Sensor-Outputsignals von der Temperatur T.

Die Auswertung der *IMU*-Daten in einer Strapdown Mechanisierung [*Titterton, 1997*] basiert auf folgendem Ansatz für die spezifische Kraft einer Komponente im körperfesten Referenzsystem (body reference system) (siehe Abb. 2.1, welche die Kräfte darstellt, die auf die seismische Masse des Beschleunigungsmessers wirken), als Funktion der linearen Beschleunigung a_x^b, der scheinbaren Zentrifugalbeschleunigung $a_{cf\_x}^b$ und der entsprechenden achsialen Komponente der statischen Gravitationsbeschleunigung g_x^b:

$$f_{m\_x} = a_x^b + a_{cf\_x}^b - g_x^b\,. \qquad (2)$$

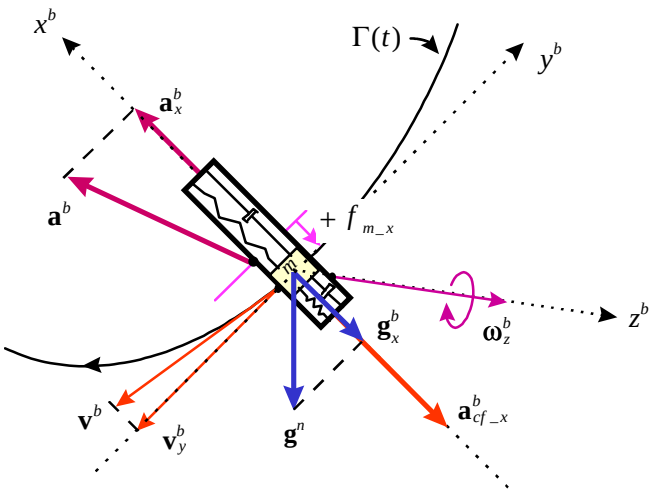


Abb. 2.1 *Spezifische Kraft als Funktion der Beschleunigungskomponenten bezüglich eines Referenzsystems, das fest mit dem bewegten Körper verbunden ist (Bsp. für die x - Achse)*

Die entsprechende vektorielle Darstellung (die spezifische Kraft sei hier mit dem Vektor **a** bezeichnet und die Korrekturterme der Zentripetal- und Gravitationsbeschleunigung seien im körperfesten System ausgedrückt) ist:

$$\mathbf{a}^b = \mathbf{a} - \boldsymbol{\omega} \times \mathbf{v}^b + \mathbf{C}_n^b \cdot \mathbf{g}^n . \qquad (3)$$

In diesem Ansatz berücksichtigen wir nur die scheinbaren Zentrifugalkräfte und die neigungsbedingten aus **g** resultierenden Beschleunigungskomponenten (als konstant angenommen). Die geringe Coriolis-Kraft, die als Konsequenz aus der Rotation des Sensorgehäuses auf die bewegte Masse wirkt, wird vernachlässigt.
Vernachlässigt werden - wie bereits erwähnt - wegen der geringen räumlichen Ausdehnung des Experiments die **g** - Variationen und wegen der mangelnden Empfindlichkeit der low-cost Sensoren der *IMU* zudem auch die aus der Erdrotation resultierende Komponente der Corioliskraft. Abb. 2.2. zeigt den Algorithmus als Flußdiagramm.

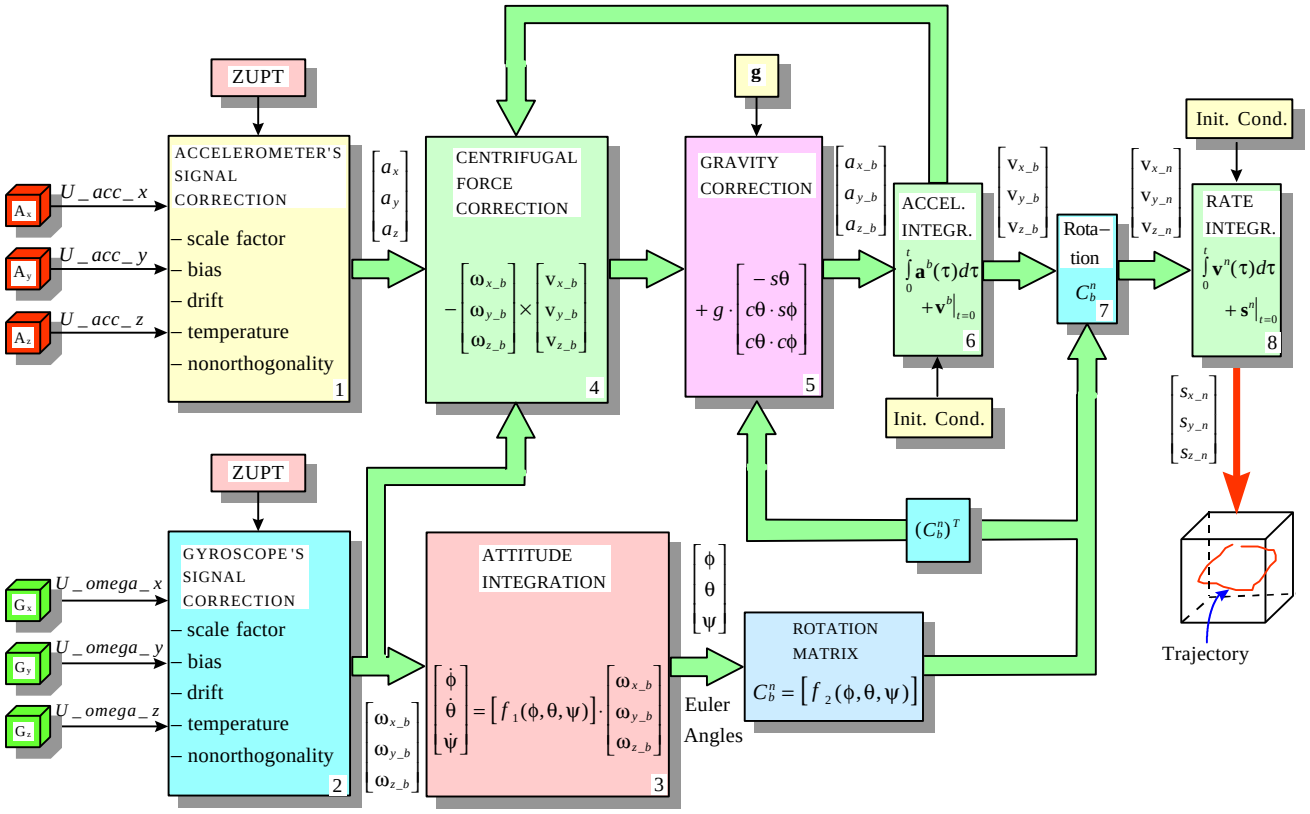


Abb. 2.2 *Flußdiagramm der Strapdown-Mechanisierung*

Er wurde als Programm in der *Simulink* - *Matlab* zugrunde liegenden Skriptsprache realisiert (siehe Abb. 2.3), die ein transparentes und interaktives Design der komplexen Signalstruktur ermöglicht sowie eine leichte und schnelle Überprüfung des Fehlermodells erlaubt.

3. Simulation einer *IMU* mit Strapdown-Mechanisierung im Labor

Bei dieser Simulation der *IMU*-Einheit mit Strapdown-Mechanisierung, die um den o.g. *ISA* von *iMAR* gebaut ist, bezieht man sich auf ein sogenanntes „Labor-Referenzsystem“ (mit “*n*” bezeichnet, Materialisierung eines Dreibeins im Testlabor), welches fest und mit einem konstanten Schwerebeschleunigungsvektor erhalten wird.
Im Fall des benutzten *ISA*-Systems basieren diese vereinfachenden Hypothesen auf einer relativ kleinen zurückgelegten Trajektorie (mit einem Umfang im Meter-Bereich, wo man eine Konstanz des Schwerebeschleunigungsvektors annehmen darf) und auf der Tatsache, daß man wegen der geringeren Auflösung der Rotationssensoren (ca. 14 °/h, im Prinzip unzureichend für das Auflösen der Rotationsbewegung der Erde von etwa $\omega_E \cong 15{,}04$ °/h) die Coriolis-Beschleunigung (die durch die Bewegung des Fahrzeugs in einem rotierenden Referenzsystem verursacht wird) vernachlässigen kann.

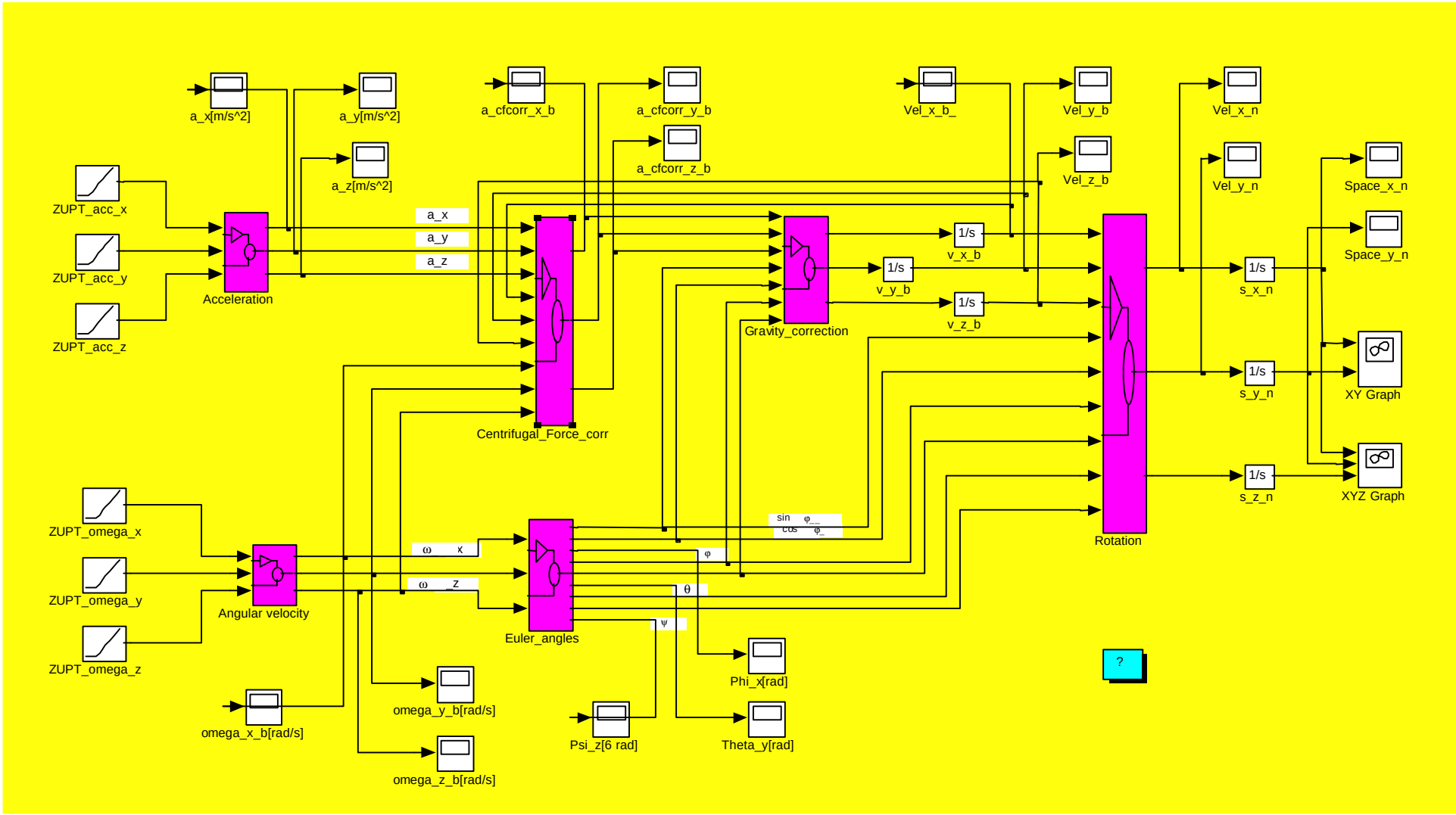


Abb. 2.3 *Simulink Programm für die Strapdown-Mechanisierung der IMU (interaktives Flußdiagramm)*

Der Zweck der Implementierung dieses Simulationsalgorithmus (siehe Abb. 2.3) ist die Abschätzung der Fehler aus der Auswertung von Daten einer bekannten Trajektorie eines Fahrzeugs. Die Daten sind die Outputs des *ISA*, welches mit dem Fahrzeug fest verbunden ist.
Man kann die Fehler durch den Vergleich mit der vordefinierten Trajektorie abschätzen und versuchen diese zu korrigieren. Es sind auch ergänzende Untersuchungen bezüglich der Stabilität und Präzision des Algorithmus möglich: durch eine theoretische Simulation der Eingangssignale, durch das Auswählen eines sinnvollen Ausführungszeitschrittes oder durch die Anwendung verschiedener Integrationsmethoden.
Zuerst erfolgte teilweise in analytischer Form ein Test unter *Maple* [*Heck, 1996; Waterloo Maple, 1997*] und danach in integraler Form unter dem graphischen Paket *Simulink* [*MathWorks Inc., 1996*], das den Vorteil eines interaktiven Operationsmodus mit Blockschaltplänen, die ausdehnbare Module enthalten, bietet. Der Simulationsalgorithmus der *IMU*-Mechanisierung wurde implementiert in 9 Hauptschritten mit 6 Hauptblöcken (Acceleration, Angular_velocity, Euler_angles, Centrifugal_Force_corr, Gravity_corr, Rotation) und einer simultanen Darstellung der Zeitvariationen einer beliebigen Größe während des Datenflußes.
Wesentliche Verbesserungen der Algorithmusstabilität für stark nichtlineare Abhängigkeiten im iterativen Rechnungsprozeß erhält man mit einer größeren Anzahl von Integrationsschritten – z. B. bei dem Runge-Kutta-Verfahren mit höherer Ordnung als 4 – und durch die Anwendung der 32 Bit Präzision in arithmetischen Operationen.
Ein geringerer Integrationsschritt bietet den Vorteil einer genaueren Integration und einer wesentlichen Verminderung der Fehler, die sich aus der Änderung der Rotationsreihenfolge im realen Fall im Vergleich zur standardmäßig angenommenen Reihenfolge ψ, θ, ϕ ergeben, und richtfertigt die Näherungen im Simulationsalgorithmus.

Zur Illustration werden einige in *Simulink* (Matlab) geschriebene Simulationsprogramme dargestellt.
Voraussetzungen sind:

"*Matlab*" ab Vers. 4.2.c.1 mit "*Simulink*" und "*Signal Processing*" Toolboxes.

In höheren Versionen als *Matlab* 4.2.c.1 oder *Simulink* 3.0 muß man entsprechende Änderungen der Programme vornehmen, um eine optimale Darstellung zu erreichen.
Die Programme können unter der Adresse **http://www.iapg.bv.tum.de/ftp/contrib/insgps/imusim.exe** heruntergeladen werden. Das selbstextrahierende zip-Archiv "imusim.exe" enthält:

- Datensätze simuliert (data*.m)
- Hilfsprogramme (vect2bf*.m) der Datenumwandlung in binärer Form (*.mat)
- Ausführbare Matlabprogramme (*.m)
- Hilfsprogramme für graphische Darstellungen (aspect.m, sfunxyz.m)

Als allgemeine Regel gilt, wenn man eventuell Änderungen anbringt, die Datei neu zu speichern oder die Session ohne Änderung in der ursprünglichen Datei zu beenden.

3.1 Translations- plus Rotationsbewegung mit simulierten Daten (Abb. 3.1, 3.2)

In Abb. 3.1 ist die berechnete Trajektorie dargestellt, die einem simulierten Datensatz entspricht (eine ebene Trajektorie, entlang der Seiten eines Quadrates, mit Translationen in der y^b – Referenzachse der *ISA* plus 90° Rotationen in den Ecken).
In Abb. 3.2 sind die berechneten Signale dargestellt, entsprechend der Trajektorie in Abb. 3.1.
Integrationsmethode: Runge – Kutta 5.
Man sieht das Ergebnis der Koordinatentransformation bei der Darstellung der Geschwindigkeiten (vgl. Abb. 3.2-d mit 3.2-e,f) und den Rotationswinkel ψ aus der Integration (im Block *Euler_angles* in Abb. 2.3) der Winkelgeschwindigkeit (Abb. 3.2 – j). In diese erste Variante des Simulink-Programms ist der Hauptblock Rotation ***bloc_rtf*** voll graphisch implementiert (vergleiche mit den späteren Varianten ***teo_trk5*** und ***teo_rrk5***, in Form einer S-Funktion geschrieben).

Befehle:
data_rt0 à 100 Hz
vect2bfn à Simulation START
bloc_rtf à Simulation START

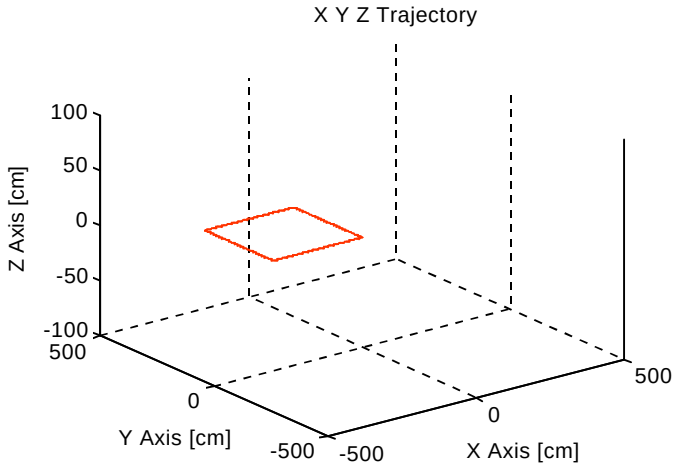
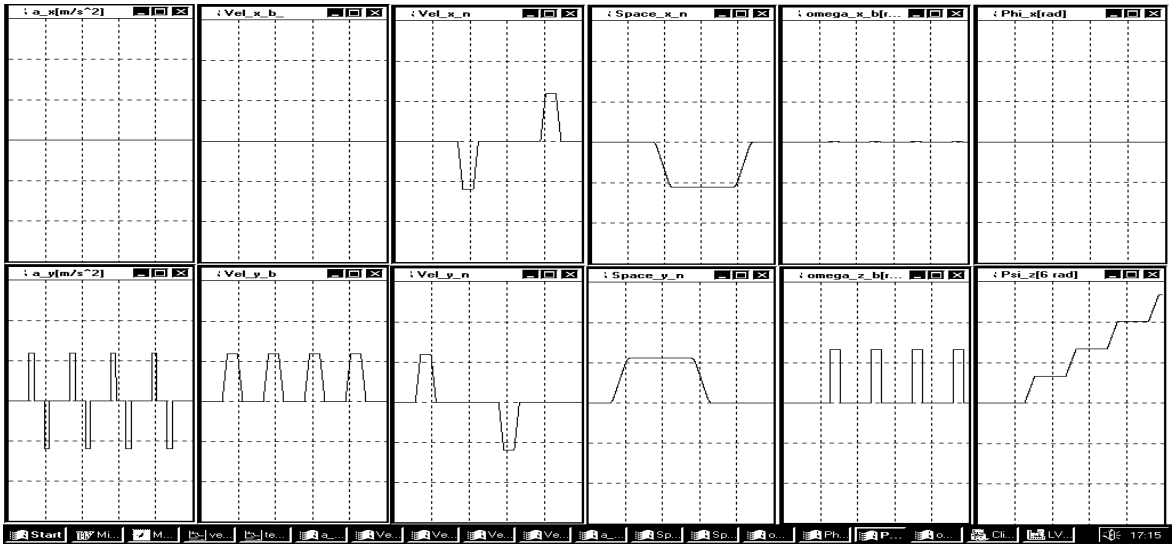


Abb. 3.1 *Simulierte rechteckige Trajektorie der Translations- plus Rotationsbewegung (Runge-Kutta 5 Integrationsmethode)*



a	c	e	g	i	k
b	d	f	h	j	l

Abb. 3.2 *Signalformen für die in Abb. 3.2 dargestellte simulierte rechteckige Trajektorie (Translations- plus Rotationsbewegung, Runge-Kutta 5 Integrationsmethode)*

3.2 Translationsbewegung, Adams-Gear Integrationsmethode, simulierte Daten (Abb. 3.3, 3.4)

In Abb. 3.3 ist die berechnete Trajektorie dargestellt, die einem simulierten Datensatz entspricht (eine ebene Trajektorie, entlang der Seiten eines Quadrates, nur mit Translationen in den x^b, y^b – Referenzachsen der *ISA*). Ab dieser Programmversion wird ein *Simulink*-S-Funktionsblock anstatt der graphischen Schaltung für den Block *Rotation* (siehe Abb. 2.3) angewandt.
In Abb. 3.4 sind die berechneten Signale dargestellt, die der Trajektorie der Abb. 3.3 entsprechen.
Integrationsmethode: Adams-Gear.
Man sieht die starke Deformation der Trajektorie (Abb. 3.3) und die entsprechenden Streckenkurven (Abb. 3.4-g,h) durch die Anwendung der Adams-Gear Integrationsmethode auch bei dieser einfachen Simulation.

Befehle:
data_tr0 à 100 Hz
vect2bfn à Simulation START
teo_trk5 à Simulation START

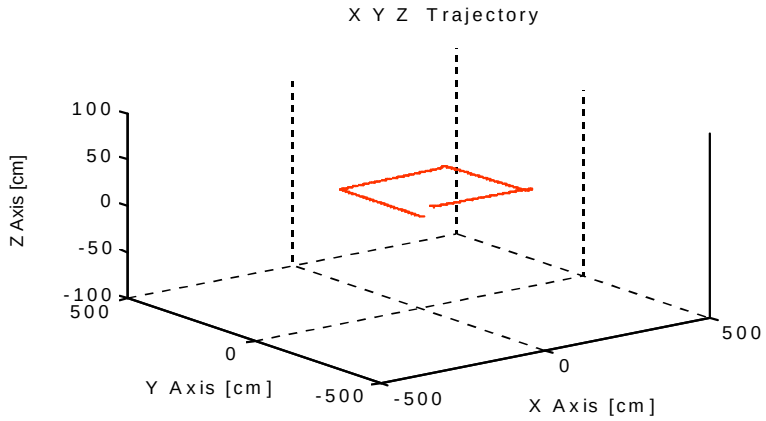
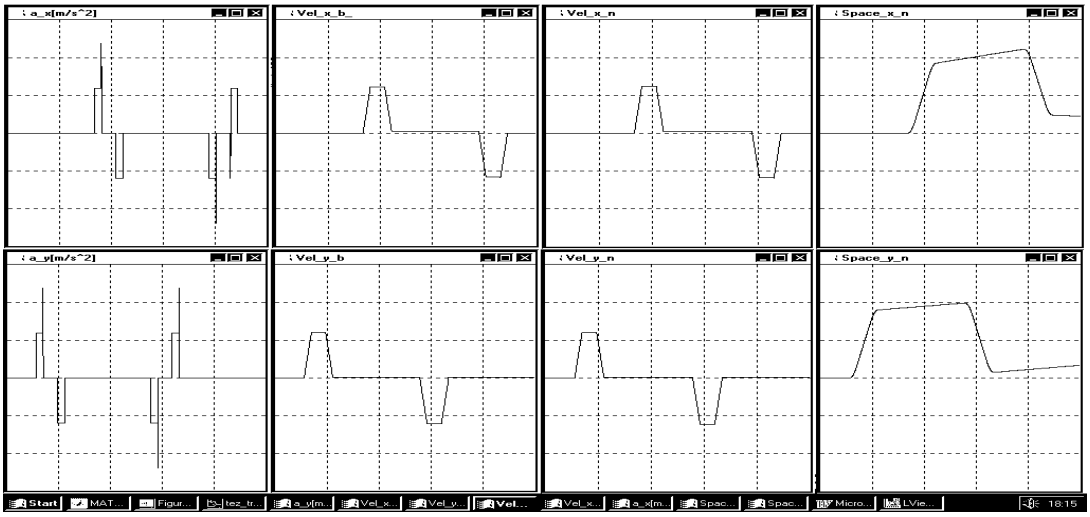


Abb. 3.3 *Simulierte rechteckige Trajektorie der Translationsbewegung (berechnet mit Adams-Gear Integrationsmethode)*



a	c	e	g
b	d	f	h

Abb. 3.4 *Signalformen für die in Abb. 3.3 dargestellte simulierte rechteckige Trajektorie (Translationsbewegung, Adams-Gear Integrationsmethode)*

3.3 Anwendung unterschiedlicher Integrationsmethoden und Integrationsschritte (ohne Abb.)

Dieselben Vergleiche können für diverse Integrationsmethoden und Integrationsschritte mit dem simulierten Datensatz für 4 aufeinanderfolgende Translationen und Rotationen durchgeführt werden.

Befehle:
data_tr0 à 100 Hz
vect2bfn à Simulation START
teo_rrk5 à Simulation START

Für den Vergleich kann man unterschiedliche Integrationsmethoden bzw. Zeitschritte testen, indem man entsprechende Änderungen in der Parameterdefinition des *Simulink* Programms vornimmt, z.B.:

à Parameters: Adams-Gear; Schritt: 10x à Simulation START
à "" Runge-Kutta 5; "" .1x à Simulation START

3.4 Translations- plus Rotationsbewegung mit realen Daten (Abb. 3.5, 3.6)

In Abb. 3.5 ist die berechnete Trajektorie dargestellt, die einem realen Datensatz entspricht (eine ebene Trajektorie, entlang der Seiten eines Quadrates, mit 4x Translationen in der y^b – Referenzachse der *ISA* plus 4x 90° Rotationen an den Ecken des Quadrats). In Abb. 3.6 sind die berechneten Signale dargestellt, entsprechend der Trajektorie in der Abb. 3.5 (Integrationsmethode: Runge-Kutta 5).

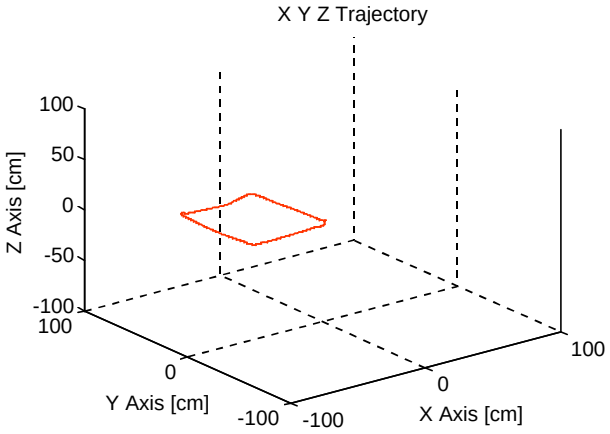
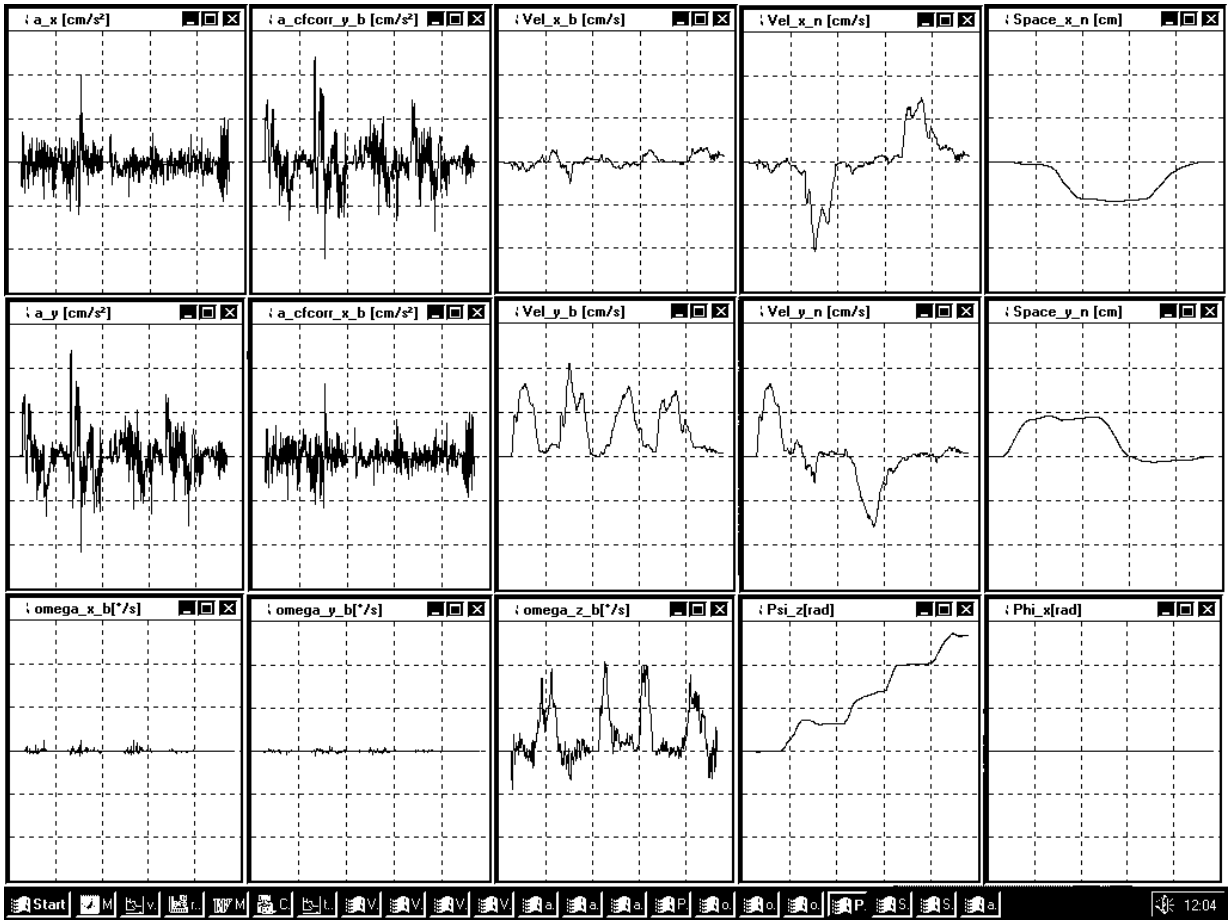


Abb. 3.5 *Reale rechteckige Trajektorie der ISA (4 aufeinanderfolgende Translationen/Rotationen; Referenzquadratseite = 60 cm – per Hand verfolgt)*



a	d	g	j	m
b	e	h	k	n
c	f	i	l	o

Abb. 3.6 *Signalformen für die in Abb. 3.5 dargestellte reale Trajektorie*

Man sieht die entsprechende Rauschreduzierung durch den Integrationsvorgang (vgl. Abb. 3.6-d, e mit 3.6-g, h (Beschleunigung / Geschwindigkeit), bzw. 3.6-i mit 3.6-l (Winkelgeschwindigkeit / Winkel)). Man sieht auch die Geschwindigkeitskomponenten-Transformation vom mobilen (*b*) in das feste Referenzsystem (*n*) (vgl. Abb. 3.6-g, h mit 3.6-j, h, wobei es im *b*-System Geschwindigkeitskomponenten nur in der *y*-Achse gibt, im Gegenteil zum *n*-System, indem es Geschwindigkeitskomponenten in der *x*- und *y*-Achse gibt).
Sehr effizient kann man Bias, Drift und Skalenfaktorfehler simulieren und deren Effekte abschätzen.

4. Schlußfolgerung

Durch Laborsimulationen einer low-cost Strapdown-*IMU* wurde in einem interaktiven Vorgang die Implementierung der Hauptschritte der Strapdown-Mechanisierung, als auch das Verhalten der Fehler der Sensoreinheit untersucht. Man konnte in diesem Versuch, wegen der niedrigeren Kreiselempfindlichkeit der low-cost *ISA* und der relativen kleinen Trajektorienmaße, die Erdrotation und die Schwerkraftänderungen vernachlässigen. Ein Ziel der Experimente ist die schnelle Abschätzung der Einflüsse unterschiedlicher Parameter (Eichungsparameter wie Skalenfaktoren, Drifts, Nichtorthogonalitäten) auf die Ausgangstrajektorie. Dabei kann man einen guten Eindruck von der Problematik der Inertialnavigation gewinnen, indem man die erwähnten Programme zur Bestimmung der Trajektorie (in der graphischen *Simulink*-Sprache geschrieben) mit simulierten oder realen Datensätze laufen läßt. Durch die simultane Darstellung aller Variablen kann man die Empfindlichkeit des Algorithmus abschätzen und die entsprechenden Korrekturen anbringen. Es ist auch möglich, die Fehler, die durch Änderungen der Rotationsreihenfolge bei den Euler-Transformationen auftreten, und die Präzision der Integration durch die Änderung des Zeitschrittes zu verdeutlichen. Im Rahmen dieses Experiments ergaben sich folgende Empfehlungen: Die Anwendung einer Abtastrate höher als 50 Hz und einer Integrationsmethode besser als Runge-Kutta 4 (siehe die Simulationsergebnisse mit der Gear Integrationsmethode von Abb. 3.3, 4, bzw. mit der Runge-Kutta 5 von Abb. 3.1, 3.2). Nach den Simulationen wurde das *Simulink*-Programm auch mit einem realen Datensatz (siehe Abb. 3.5, 6) mit befriedigenden Ergebnissen ausgeführt; die linearen Driftkomponenten der Trägheitssensoren wurden mit Hilfe der Ramp-Funktion ausgeglichen (erst unter *Simulink* Ver. 2 verfügbar).
Als nächster Schritt bietet sich das Vervollständigen des Algorithmus für die reale Navigation an, wobei auch Schwerefeldeinflüße und Corioliskraft (wegen der Erdrotation) bei der Anwendung hochwertiger *ISA*-Einheiten berücksichtigt werden sollten; die Quaternion-Parametrisierung für die 3D-Navigationslösung wäre zu empfehlen, um das Singularitätsproblem zu vermeiden und um Recheneffizienz zu erhöhen.
Das *Simulink*-Programm kann man auch als Kern für ein 2D Real-time Navigationsprogramm verwenden; mittels des Real-Time Workshops (*MathWorks, Inc.*) und eines entsprechenden C-Compilers führt man auf einem *DSP* Processor-Board den von *Simulink* in *DSP*-Maschinensprache umgewandelten Code aus.
Im Anhang sind einige Schaltblöcke des Programms genauer erklärt; der Programm-Quelltext, als auch Simulationsdatensätze stehen zur Verfügung unter die *HTML* Adresse:
http://www.iapg.bv.tum.de/ftp/contrib/insgps/imusim.exe

Anerkennung

Ich bedanke mich bei Herrn Prof. R. Rummel für die wertvollen Anregungen bei der Durchführung dieser Arbeit. Für die fruchtbar geführten Diskussionen gilt weiter mein Dank den Herren Prof. M. Schneider, Ir. N. Sneeuw, Dipl.-Ing. B. Zebhauser und Dr. J. Müller.

5. Literatur

Britting, K. R. (1971): „*Inertial Navigation Systems Analysis*“, Wiley - Interscience, New York

Heck, A. (1996): „*Introduction to Maple*“, 2-nd Ed., Springer, New York

Heinze, O. (1996): "*Aufbau eines operablen inertialen Vermessungssystems zur Online-Verarbeitung in der Geodäsie auf Basis eines kommerziellen Strapdown Inertialsystems*", Dissertation, Reihe C, Heft Nr. 459, Verlag der Bayerischen Akademie der Wissenschaften, München

iMAR (1995): "iMARtgac - *Installationshinweise, Korrekturmodell und Kalibrierung*" (Gesellschaft für Mess-, Automatisierungs- und Regelsysteme)

Kayton, M., Walter R. F. (1997): „*Avionics Navigation Systems*“, 2-nd Ed., John Wiley & Sons Inc., New York

MathWorks, Inc. (1996): *MATLAB Vers. 4.2.c (Control System Toolbox, Signal Processing Toolbox Vers. 4.0, Simulink Vers. 2.0, Statistics Toolbox)*

National Instruments (1995): „*DAQPad -MIO-16XE-50, 16-Bit Data Acquisition and Control for the Parallel Port*“, User Manual, Austin

Nelson, R. C. (1996): „*Flight Stability and Automatic Control*“, 2-nd Ed., McGraw-Hill, New York

Rummel, R. (1986): „*Inertial Surveying*“, Technische Hegeschool Delft

Schmidt, G. T. (1978): „*Strapdown Inertial Systems - Introduction and Owerview*“, AGARD (Advisory Group for Aerospace Research and Development) Lecture Series No. 95

Titterton, D., H., Weston, J., L. (1997): „*Strapdown inertial navigation technology*“, IEE Books, Peter Peregrinus Ltd., UK

Waterloo Maple Inc. (1997): „*Maple V Release 5.2“*

ANHANG

Beispiele für die Zuordnung eines Formelsatzes zum *SIMULINK*-Blockschaltbild

Im folgenden werden Beispiele für die Implementierung einiger Funktionalblöcke aus dem allgemeinen Rechenblock der Strapdown-Mechanisierung der *IMU* (siehe Abb. 2.3) präsentiert. Es wird auch eine Möglichkeit der binären Datenvorbereitung als Input für das interaktive *Simulink* Programm erläutert, die im realen Fall von dem A/D Konverter generiert werden (siehe Abb. A.5).

A.1 Kalibrierungs- und Bias- (und Drifts-) Korrektionsblock für die Beschleunigungssensorsignale.

$$\begin{bmatrix} a\_x \\ a\_y \\ a\_z \end{bmatrix} = \left(\left(1000\, g_{iMAR} \begin{bmatrix} k\_acc\_xx & k\_acc\_yx & k\_acc\_zx \\ k\_acc\_xy & k\_acc\_yy & k\_acc\_zy \\ k\_acc\_xz & k\_acc\_yz & k\_acc\_zz \end{bmatrix} \right) \&* \begin{bmatrix} \frac{1}{skf\_acc\_x} & 0 & 0 \\ 0 & \frac{1}{skf\_acc\_y} & 0 \\ 0 & 0 & \frac{1}{skf\_acc\_z} \end{bmatrix} \right) \&* \begin{bmatrix} U\_acc\_x - U\_Offset\_acc\_x - U\_bias\_acc\_x\_ZUPT - U\_drift\_acc\_x\_ZUPT \\ U\_acc\_y - U\_Offset\_acc\_y - U\_bias\_acc\_y\_ZUPT - U\_drift\_acc\_y\_ZUPT \\ U\_acc\_z - U\_Offset\_acc\_z - U\_bias\_acc\_z\_ZUPT - U\_drift\_acc\_z\_ZUPT \end{bmatrix}$$

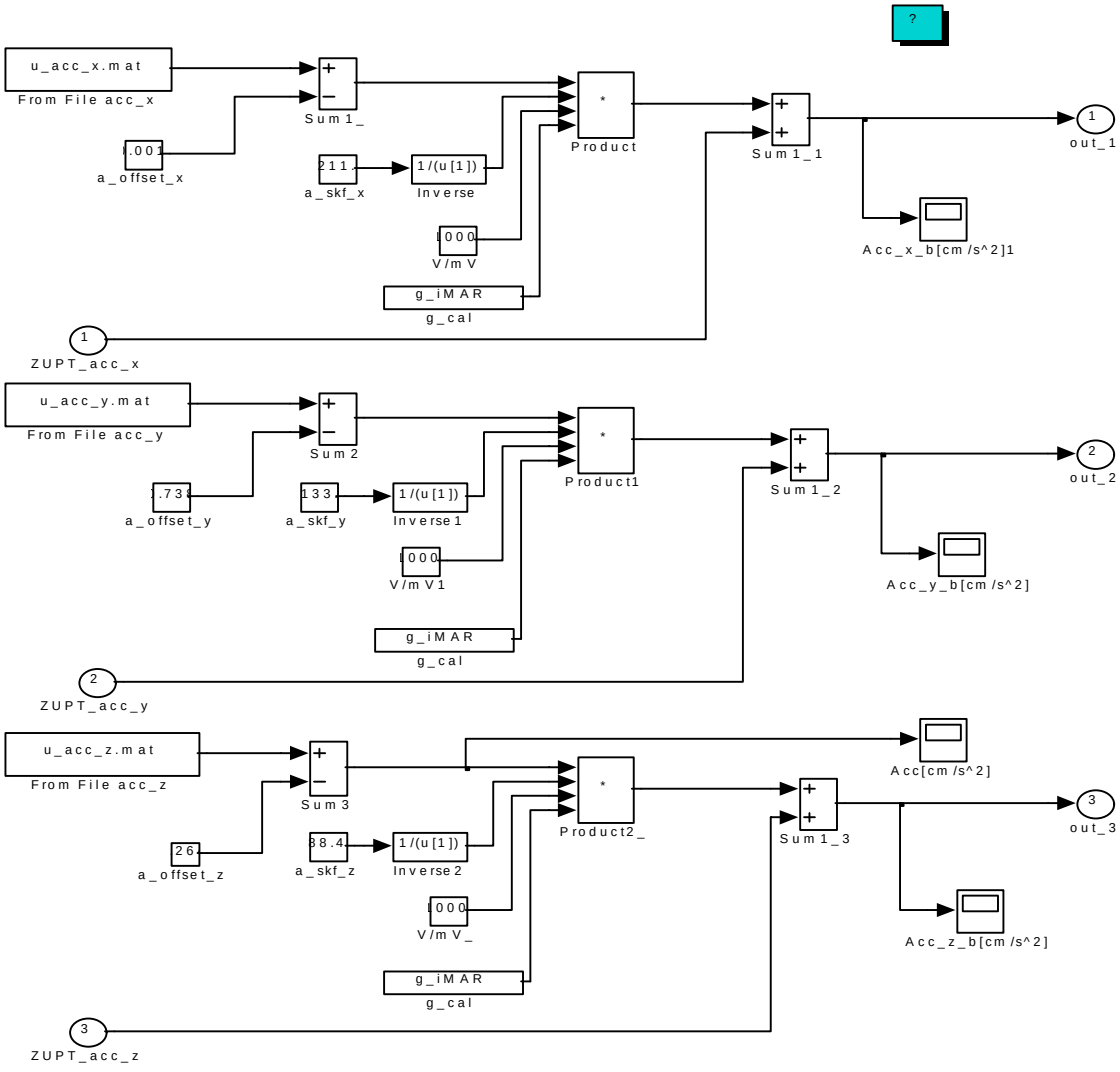


Abb. A-1 *Implementierung des Blocks 1 (Accelerometer Signal Correction & Calibration) des Simulationsalgorithmus (siehe Abb. 2.3)*

A.2 Integrationsblock für die Bestimmung der Euler-Rotationswinkel [*Nelson, 1996*]

$$\begin{bmatrix} \frac{\partial}{\partial t}\phi \\ \frac{\partial}{\partial t}\theta \\ \frac{\partial}{\partial t}\psi \end{bmatrix} = \begin{bmatrix} 1 & \frac{\sin(\theta)\sin(\phi)}{\cos(\theta)} & \frac{\sin(\theta)\cos(\phi)}{\cos(\theta)} \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} * \begin{bmatrix} \omega_{x\_b} \\ \omega_{y\_b} \\ \omega_{z\_b} \end{bmatrix}$$

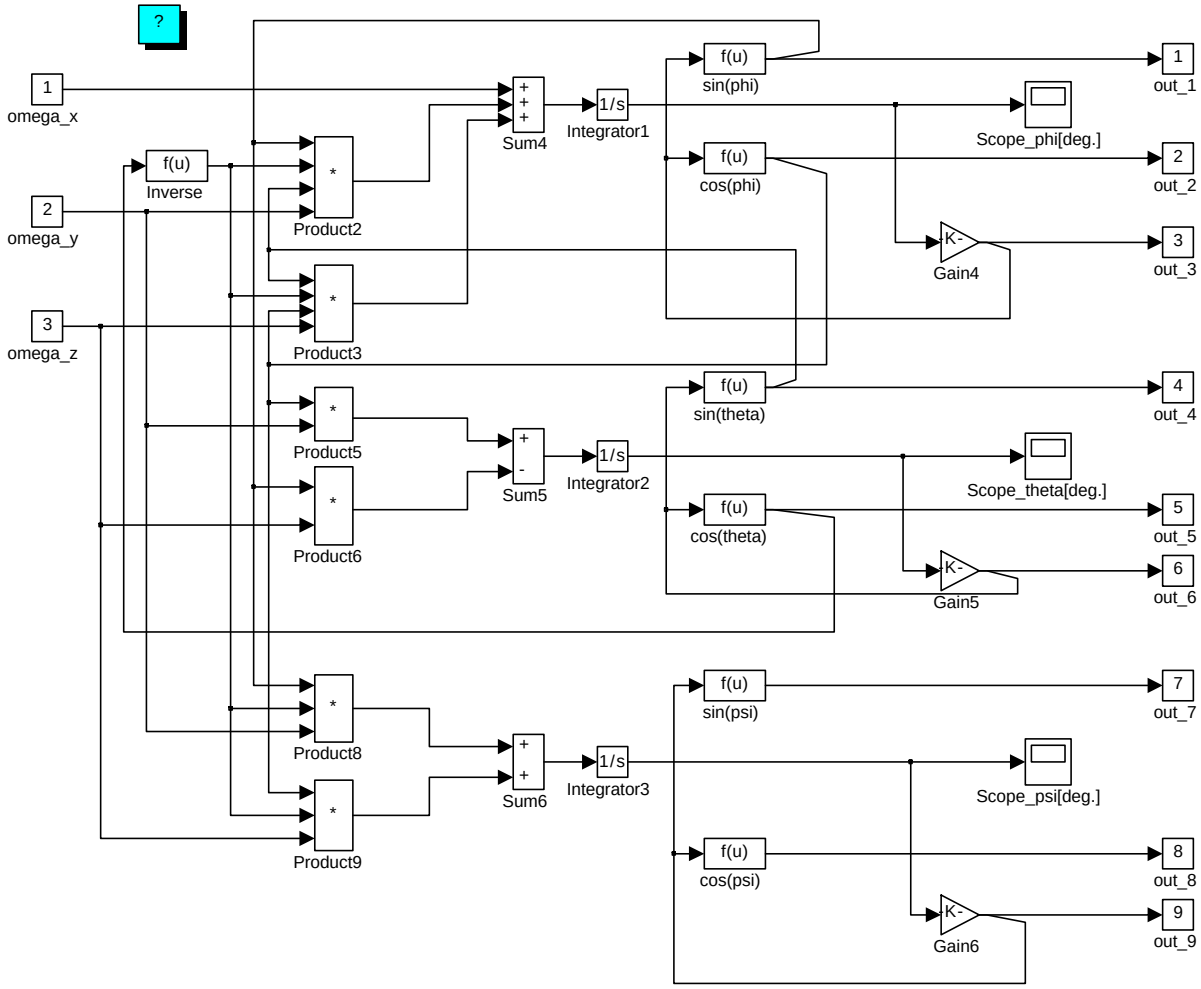


Abb. A-2 *Implementierung des Blocks **3** (Attitude Integration) des Simulationsalgorithmuess (siehe Abb. 2.3)*

A.3 Vektorkomponententransformation vom *b* (body) zum *n* (als fest betrachtet) Referenzsystem

$$\begin{bmatrix} v\_x\_n \\ v\_y\_n \\ v\_z\_n \end{bmatrix} = \begin{bmatrix} \cos(\psi)\cos(\theta) & -\sin(\psi)\cos(\phi)+\cos(\psi)\sin(\theta)\sin(\phi) & \sin(\psi)\sin(\phi)+\cos(\psi)\sin(\theta)\cos(\phi) \\ \sin(\psi)\cos(\theta) & \cos(\psi)\cos(\phi)+\sin(\psi)\sin(\theta)\sin(\phi) & -\cos(\psi)\sin(\phi)+\sin(\psi)\sin(\theta)\cos(\phi) \\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \&* \begin{bmatrix} v\_x\_b \\ v\_y\_b \\ v\_z\_b \end{bmatrix}$$

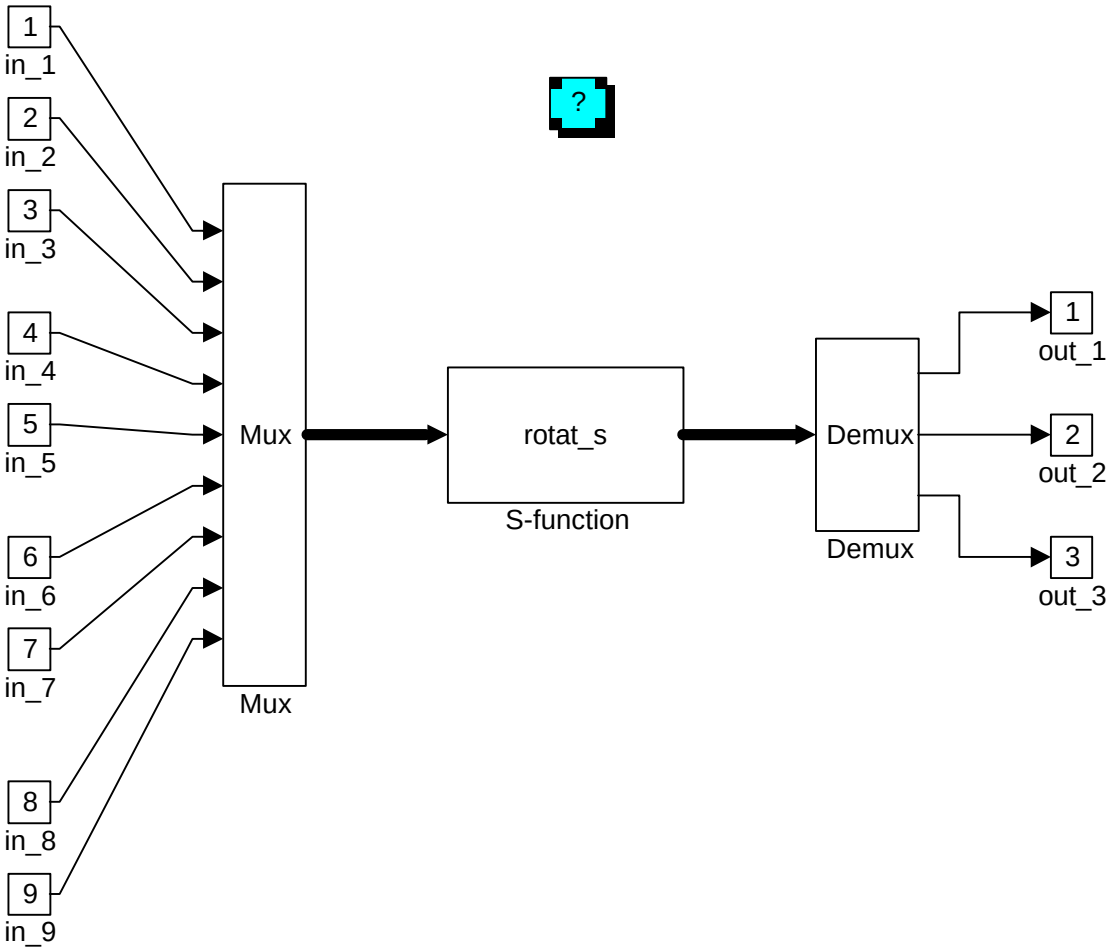


Abb. A-3 *Implementierung des Blocks **6** (Rotation) des Simulationsalgorithmus (siehe Abb. 2.3)*

A.4 Beispiel eines S-Funktions-Programms (Matlab-Skript für einen *Simulink* Ausführungsblock)

Neben der rein graphischen Programmierung in *Simulink* ist manchmal die Anwendung einer Kombination von analytischer und graphischer Programmierung vorteilhaft. In der Abb. A 4 ist die Funktion *Rotationsmatrix* in analytischer Form in *Matlab*-Syntax angegeben. Das Programm ist in einem graphischen *Simulink*-Block der S-Functions-Kategorie implementiert. Man sieht die Notwendigkeit vektorieller Eingänge und Ausgänge, die mit Multiplexer- / Demultiplexer-Blöcken zu skalaren Komponenten gebildet werden.
Sinuse oder Kosinuse der entsprechenden Eulerwinkel (6 Werte) und die Komponenten des Geschwindigkeitsvektors im *b* – Referenzsystem (3 Werte) sind Eingangsgrößen; als Ausgangsgrößen erhält man die Geschwindigkeitskomponenten im *n* – Referenzsystem (3 Werte).

```
%###############  begin rotmult.m (Rotation - multivariable) ########

function [sys, x0] = rotmult (t, x, u, flag)

if abs (flag) == 0,

        sys = [0 0 3 9 0 0];
        x0 = 0;

elseif flag == 3,

        sys(1) = u(9)*u(7)*u(1)+(-u(8)*u(5)+u(9)*u(6)*u(4))*u(2)+...
                                  (u(8)*u(4) + u(9)*u(6)*u(5))*u(3)

        sys(2) = u(9)*u(7)*u(1)+ (u(9)*u(5)+u(8)*u(6)*u(4))*u(2)+...
                                   (-u(9)*u(4) + u(8)*u(6)*u(5))*u(3)

        sys(3) =     -u(6)*u(1) +              u(7)*u(4)*u(2)+...
                                                  u(7)*u(5)*u(3)

else

        sys = [];

end

%############## end rotmult.m ###################################

% test
% [sizes, x0] = rotmult( [], [], [], 0 )
% sys(3) = rotmult([], [], [], 3 )
% out = rotmult( [], [], [1 2 3 4 5 6 7 8 9], 3 )
```

Abb. A-4 *Ausdruck der „rotat_s“ Funktion des Blocks **6** (Rotation) (siehe Abb. A-3)*

A.5 Datenvorbereitungsblock (vom ASCII zum binären Format)

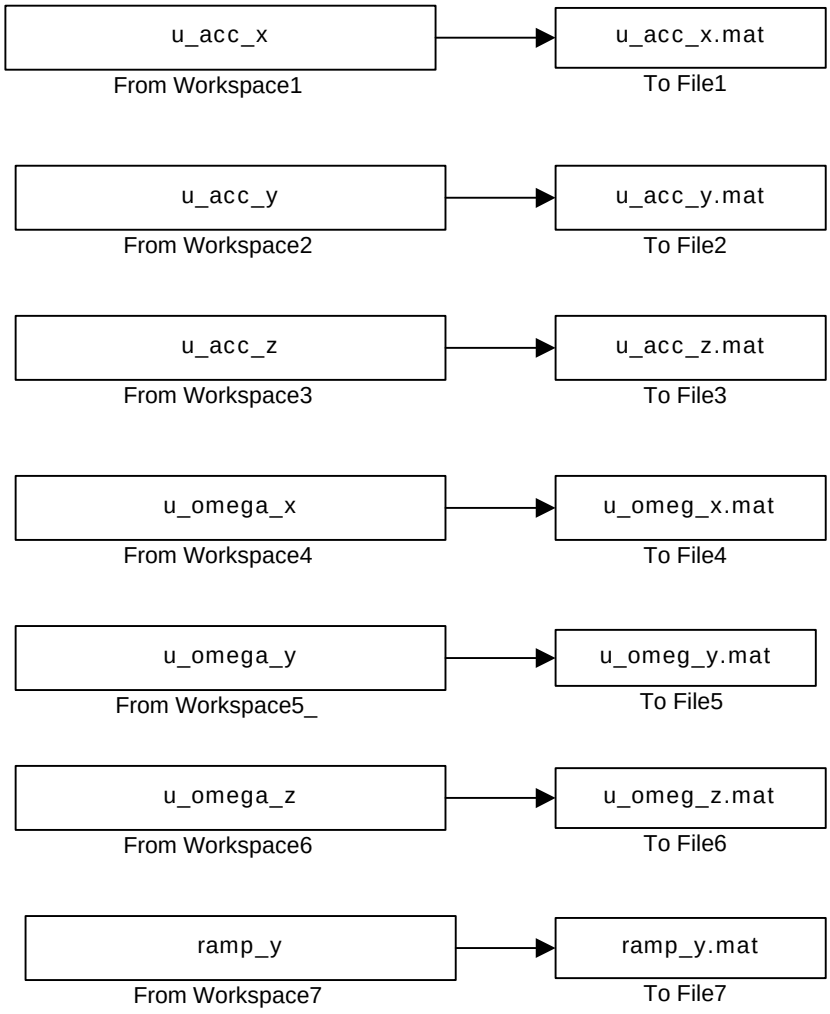


Abb. A-5 *Simulinkblock (**vect2bfn**) für die Umwandlung der registrierten ASCII- in binäre Daten, wie sie vom Simulationsalgorithmus benötigt werden*