CES Softwareentwicklungspraktikum

Analyse- und Entwurfsdokument

Lena Blum, Alexander Fischer und William Hulin

Matr.-Nr. 302253, 303979 und 293858
email:
[lena.blum|alexander.fischer|william.hulin]@rwth-aachen.de

Inhaltsverzeichnis

Kapitel 1

Vorwort

1.1 Aufgabenstellung und Struktur des Dokuments

Aufgabenstellung

Im Rahmen des Softwareentwicklungspraktikums (CES_SS2012) soll eine Software zur Simulation eines Stehaufkreisels erstellt werden. Die Simulationssoftware muss sowohl den reibungsfreien, als auch den reibungsbehafteten Fall korrekt simulieren können.
Als Programmiersprache soll C++ verwendet werden. Der Quellcode soll derart strukturiert und kommentiert sein, dass spätere Modifikationen und Erweiterungen durch Dritte möglich sind.

1.2 Projektmanagement

Protoyping (MATLAB/ FORTRAN)	Alexander
Dokumentation	Lena
Coding:	
Parameterset, Solver, Solution, Rkv56Parset, Rkv56, DESolution,«interface»RightSide, RHS, Rkv56Modified	Alexander
«interface »OutputInterface, OutputToolbox, Main, ExceptionHandlingModule, MathException, NonCriticalME, CriticalME, ParameterException	William
GUI	Lena

Kapitel 2

Analyse

2.1 Anforderungsanalyse

2.1.1 Benutzeranforderungen

Das von Herrn Professor Gauger gestellte Simulationsproblem umfasst die Erstellung einer Software zur Simulation eines Stehaufkreisels.
Die Simulation muss sowohl den reibungsbehafteten, als auch reibungsfreien Fall korrekt simulieren.
Im Speziellen wird ein Runge-Kutta56-Verfahren mit adaptiver Schrittweitensteuerung unter Betrachtung einer Erhaltungsgröße (*conserved quantity*) zur Simulation des Problems verwendet.

Das Rkv56 Verfahren wurde durch ein StepperDopr853-Verfahren ersetzt, um eine höhere Genauigkeit zu erreichen.
Die Realisierung der Simulation findet in C++ statt.

Die Bedienung sowie das Ausgeben der Simulationsergebnisse muss durch eine grafische Benutzeroberfläche (*GUI*) möglich sein.
Die Simulationsergebnisse können in einer *ASCII*-formatierten Datei zur weiteren Verarbeitung und Auswertung exportiert werden.
Durch den modularen Aufbau ist die Wartbarkeit und einfache Erweiterbarkeit der Software durch Dritte gewährleistet.
Das Kernproblem besteht im Lösen der Rechten Seite des folgenden Differentialgleichungssystems:

$$\begin{aligned}
&\ddot{\theta}(I + ma^2 \sin^2\theta + kma \sin\theta(R - a\cos\theta)(-\dot{x}_c \sin\phi + \dot{y}_c \cos\phi + (R - a\cos\theta)\dot{\theta})) \\
&= \underbrace{-(I_3 - I)\dot{\phi}^2 \sin\theta\cos\theta}_{=0} - I_3\dot{\phi}\sin\theta\dot{\psi} + (g + a\dot{\theta}^2\cos\theta)(-ma\sin\theta - km(R - a\cos\theta) \\
&(-\dot{x}_c \sin\phi + \dot{y}_c \cos\phi + (R - a\cos\theta)\dot{\theta}))
\end{aligned}$$

$$\ddot{\phi} I \sin\theta = -\underbrace{(2I - I_3)}_{=I}\dot{\phi}\dot{\theta}\cos\theta + I_3\dot{\theta}\dot{\psi}$$
$$- km(g + a\cos\theta\dot{\theta}^2 + a\sin\theta\ddot{\theta})(a - R\cos\theta)(\dot{x}_c\cos\phi + \dot{y}_c\sin\phi + (a\dot{\phi} + \dot{\psi}R)\sin\theta)$$

$$\ddot{\psi} I_3 = -I_3(\ddot{\phi}\cos\theta - \dot{\phi}\dot{\theta}\sin\theta)$$
$$- km(g + a\cos\theta\dot{\theta}^2 + a\sin\theta\ddot{\theta})(R\sin\theta)(\dot{x}_c\cos\phi + \dot{y}_c\sin\phi + (a\dot{\phi} + \dot{\psi}R)\sin\theta)$$

$$m\ddot{x}_c = -km(g + a\cos\theta\dot{\theta}^2 + a\sin\theta\ddot{\theta})(\dot{x}_c + (a\dot{\phi} + \dot{\psi}R)\sin\theta\cos\phi + (a\cos\theta - R)\sin\phi\dot{\theta})$$

$$m\ddot{y}_c = -km(g + a\cos\theta\dot{\theta}^2 + a\sin\theta\ddot{\theta})(\dot{y}_c + (a\dot{\phi} + \dot{\psi}R)\sin\theta\sin\phi + (R - a\cos\theta)\cos\phi\dot{\theta})$$

2.1.2 Anwendungsfallanalyse

Beschreibung der Anwendungsfälle

Name	Export as Tecplot file	
Ziel	Enable storage of simulation data in a common file format	
Einordnung		
Vorbedingung	Simulation has been run or imported	
Nachbedingung	A tecplot file has been created	
Nachbedingung im Fehlerfall		
Haupt-Neben-akteure	User	
Auslöser	User presses the *Export Data* button	
Standardfluss	**Schritt**	**Aktion**
	1	User presses the *Export Data* button
	2	User choses a file name and directory in a new dialog
	3	Export file is created and the main window resumed

Name	Import from Tecplot file	
Ziel	Enable recovery of previously stored data	
Einordnung		
Vorbedingung	Simulation data has been exported as a tecplot file	
Nachbedingung	Simulation data is loaded and graphed	
Nachbedingung im Fehlerfall		
Haupt-Neben-akteure	User	
Auslöser	User presses the *Import Data* button	
Standardfluss	**Schritt**	**Aktion**
	1	User presses the *Import Data* button
	2	User selects a file to import from a new dialog
	3	Simulation data is loaded and graphed in the main window

Name	Input Parameters	
Ziel	A set of valid parameters has been loaded into memory	
Einordnung		
Vorbedingung		
Nachbedingung	The user can start a simulation	
Nachbedingung im Fehlerfall	Errormessage is shown	
Haupt-Neben-akteure	User	
Auslöser	User presses the *Change Parameter* button in the GUI	
Standardfluss	**Schritt**	**Aktion**
	1	User presses the *Change Parameter* button in the GUI
	2	User enters parameters in a popup-window or clicks on the *Import Parameters* button and selects a file in the new dialog
	3	User clicks *Submit Changes* to close the parameters dialog
Nebenfluss	**Schritt** 2a	**Aktion** User can export the current parameters to a file using the *Export Parameters* button

Save/Load parameters from file: See *Input Parameters*.

Name	Start Simulation	
Ziel	Run the mathematical solver with the given parameters	
Einordnung		
Vorbedingung	Parameters have been entered and checked for validity	
Nachbedingung	Solver is finished, output is being created	
Nachbedingung im Fehlerfall	Solver could not finish calculation, exception dialog is displayed	
Haupt-Neben-akteure	User, Solver	
Auslöser	User presses the *Simulate* button in the GUI	
Standardfluss	**Schritt**	**Aktion**
	1	User starts the simulation
	2	GUI is disabled
	3	Run the solver
	4	Draw output Graphs
	5	Enable GUI

Name	Toggle View	
Ziel	Enable the user to view a different set of graphs	
Einordnung		
Vorbedingung	There is simulation data to be graphed	
Nachbedingung	A different graph is displayed	
Nachbedingung im Fehlerfall		
Haupt-Neben-akteure	User	
Auslöser	User toggles the spinbox	
Standardfluss	**Schritt**	**Aktion**
	1	User selects a different graph from the spinbox
	2	A new set of data is graphed according to the number selected

Systemanforderungen

Funktionale Anforderungen

Dem Anwender ist es möglich die Simulationsparameter k (Reibung) sowie $\dot{\psi}(rad/s), \theta(rad), R(cm), a(cm), m(g)$ und die Toleranz der Erhaltungsgröße über eine grafische Eingabemaske festzulegen. Wenn während der Simulation ein Fehler auftritt wird der Anwender über ein Popup-Fenster benachrichtigt. Nach Durchlauf der Simulation bekommt der Anwender die Simulationsergebnisse - $\theta, \psi, \phi, x_c, y_c, \dot{\theta}, \dot{\psi}, \dot{\phi}, v_x, v_y$ - in Form von *LineCharts* in eine *GUI* eingebettet angezeigt.
Die auf der *GUI* ausgegebenen Plots können als Bilddatei oder im Tecplotformat exportiert werden.

Kommt es während der Laufzeit zu einem kritischen Fehler (ein Fehler, der das korrekte Fortführen des Programmes unmöglich macht) wird der Anwender über

ein Popup-Fenster benachrichtigt und das an die Stelle zurückgesetzt, an der der Fehler auftrat.

Nicht-Funktionale Anforderungen

Die Exportfunktion der Simulationssoftware schreibt Tecplot konforme ASCII-kodierte Ausgabedateien. Vormals exportierte Dateien können wieder importiert und geplottet werden. Ebenso können ältere Parameterkonfigurationen importiert werden.

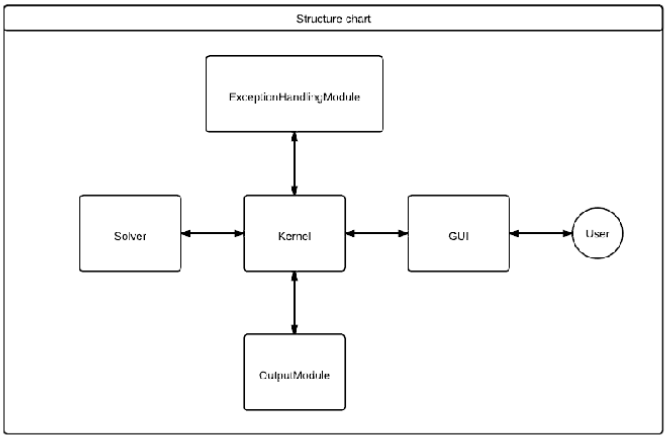
2.2 Begriffsanalyse

- LineChart - *Zwei Achsen Diagramm mit Kartesischem Koordinatensystem. Die einzelnen Datenpunkte sind durch gerade Linien verbunden.*
- GUI - *Eine grafische Benutzeroberfläche (GBO oder GUI) ist eine Software-Komponente, die dem Benutzer eines Computers die Interaktion mit der Maschine über grafische Symbole erlaubt.*
- θ - *Nutation*
- ϕ - *Präzession*
- ψ - *Rotation*
- x_c - *x-Koordinate*
- y_c - *y-Koordintate*
- $\dot{\theta}$ - *Nutationsgeschwindigkeit*
- $\dot{\phi}$ - *Präzessionsgeschwindigkeit*
- $\dot{\psi}$ - *Rotationsgeschwindigkeit*
- v_x - *Geschwindigkeit in x-Richtung*
- v_y - *Geschwindigkeit in y-Richtung*
- R - *Radius*
- k - *Reibungskoeffizient*
- a - *Abstand vom Mittelpunkt zum Schwerpunkt*
- m - *Masse des Kreisels*
- G - *Erhaltungsgröße*
- *atol* - *absolute Toleranz des Runge-Kutta-Verfahrens*
- *rtol* - *relative Toleranz des Runge-Kutta-Verfahrens*

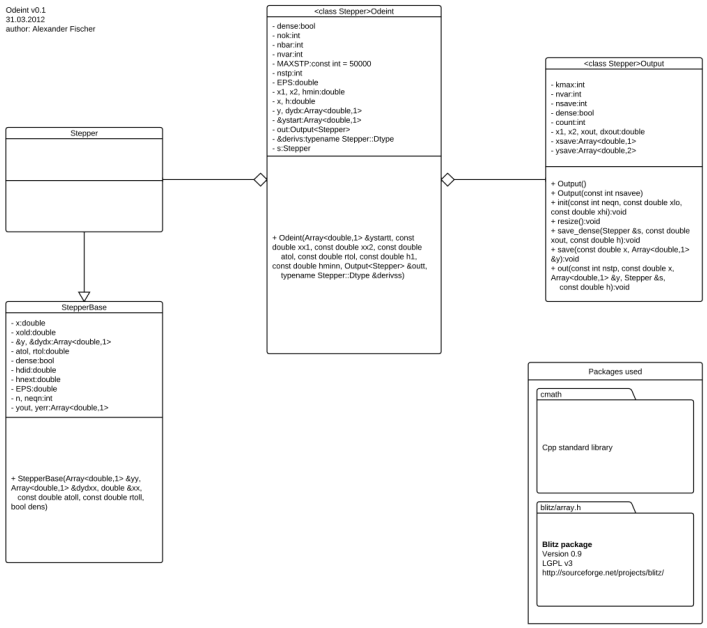
Kapitel 3

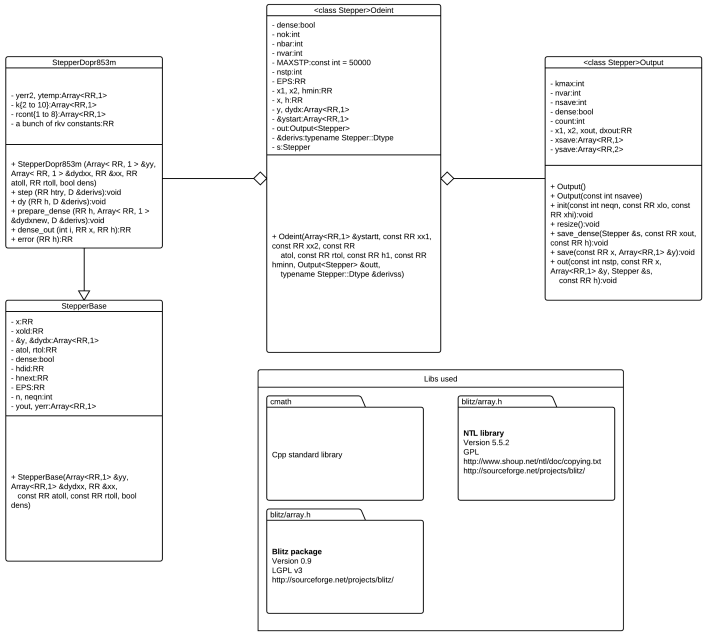
Entwurf

3.1 Grobentwurf: Subsysteme



3.2 Detailentwurf: Klassen





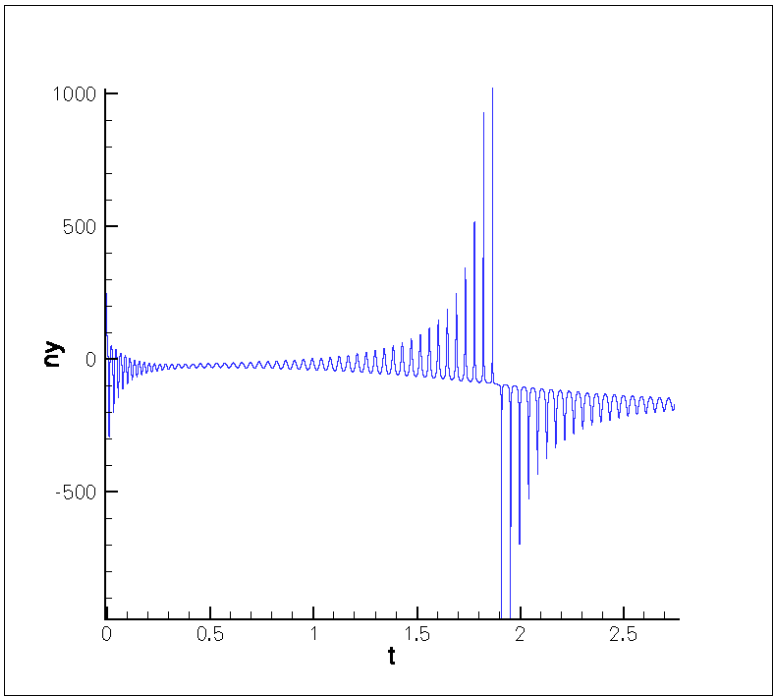
StepperDopr853 - Dormand-Prince 853 method

Entwicklungsschritte vom Prototypen zum fertigen Löser
Um Referenzdaten erzeugen zu können und frühzeitig mathematische Fehler ausschließen zu können haben wir uns für die Implementierung eines Prototypen entschieden. Nach der ersten Implementierung eines rkv56 Verfahrens in Matlab entschieden wir uns, zu Gunsten einer höheren Genauigkeit und Geschwindigkeit, weiter Implementierungen in Fortran95 zu programmieren.
Der fertige Fortran Prototyp, ebenfalls ein Runge-Kutta 56 mit adaptiver Schrittweitensteuerung, benötigte für die Lösung des TippeTop Problems[1]

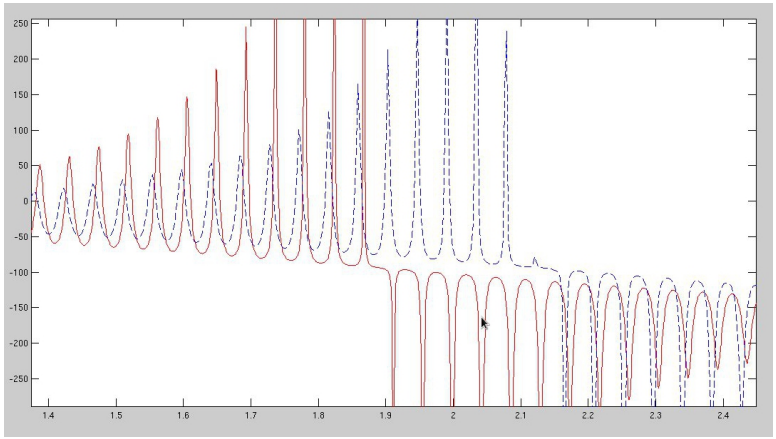
```
Simulation of the TippeTop gyro
Performing rkv56
Solution computed
Steps:
 3160922
Done
```

Lösung des Prototypen für $\dot{\psi}$ in rot/s

[1] $k = 0.3, h_{min} = 10^{-8}, h_{max} = 10^{-6}, rtol = atol = 10^{-4}, y0 = (0.0, 0.0, 250.0, 0.0, 0.0, 0.1, 0.0, 0.0, 0.0, 0.0), T = [0, 2.75], dG < 10^{-6}, 3160922$ Einzelschritte.



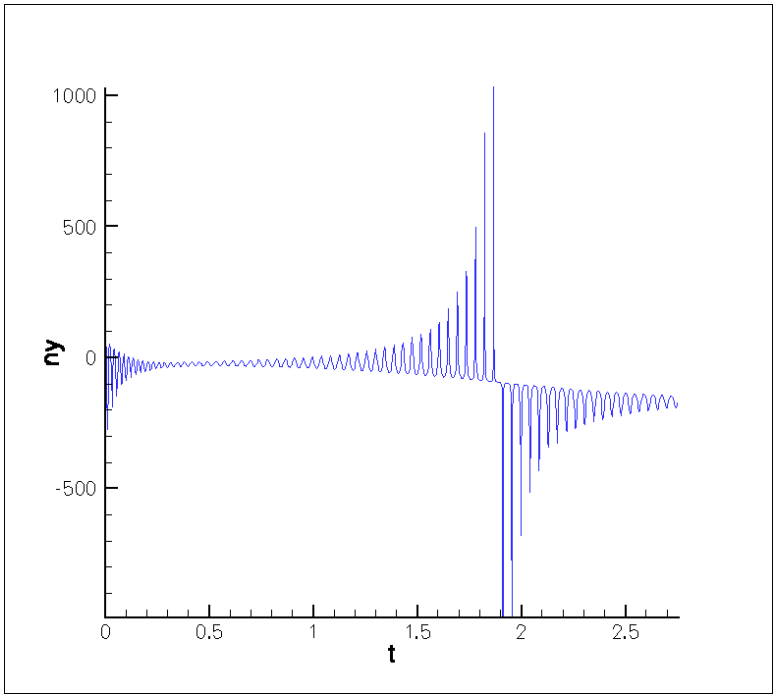
Auf Grund der Erfahrung mit dem Prototypen entschieden wir uns für die Verwendung eines Dopr853 Verfahrens. Die erste Implementierung unter Verwendung des Datentyps double (auf 15 Nachkommastellen genau) lieferte leider signifikant falsche Ergebnisse



Die korrekte Lösung der rkf56 Fortran95 Implementierung ist rot dargestellt, die falsche Lösung des Dopr853_double Algorithmus in blau. Es lag nahe das diese Unterschiede in der Lösung auf Ungenauigkeiten in der Auswertung der steifen rechten Seite und den Berechnungen des Dopr853 Algorithmus zurückzuführen waren. Wir entschieden uns also für die Verwendung eines genaueren Datentyps,

und zwar NTL::RR aus der NTL library[2]
Unter verwendung des NTL::RR Datentyps kann der fertige Löser (Dopr853 in *C++*) die Lösung des Problems[1] in 852 Schritten berechnen.

Lösung für $\dot{\psi}$(Drehgeschwindigkeit) mit Dopr853:



Der Dopr853 Algorithmus ist ein Algorithmus aus der Familie der Runge-Kutta Algorithmen der Ordnung 8. Für jeden Schritt werden 12 Auswertungen der rechten Seite des DGL benötigt. Der ursprüngliche Algorithmus nutzte eine Fehlerschätzung der Ordnung 6, was sich allerdings in einigen Fällen als unzureichend herausstellte, da dieser Fehlerschätzer jeweils die letzte Auswertung nicht berücksichtigte. Hairer, Nörsett und Wanner[3] konstruierten Abschätzungen der fünften und dritten Ordnung, die auch den letzen Punkt berücksichtigen.
Der Fehler kann also über
$err = err_5 \frac{err_5}{\sqrt{(err_3)^2 + 0.01(err_5)^2}}$
abgeschätzt werden.
Die meiste Zeit über gilt $err_5 << err_3$ und damit $err = O(h^8)$.

StepperDopr853 wurde als Mehrschrittverfahren mit fehlergesteuerter Schrittweitensteuerung implementiert, die neben dem geschätzten Fehler auch noch die Erhaltungsgröße

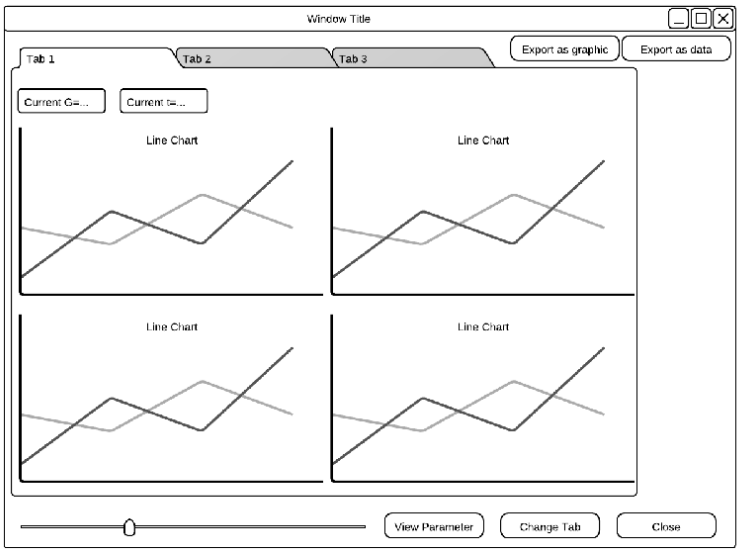
[2] http://www.shoup.net/ntl/

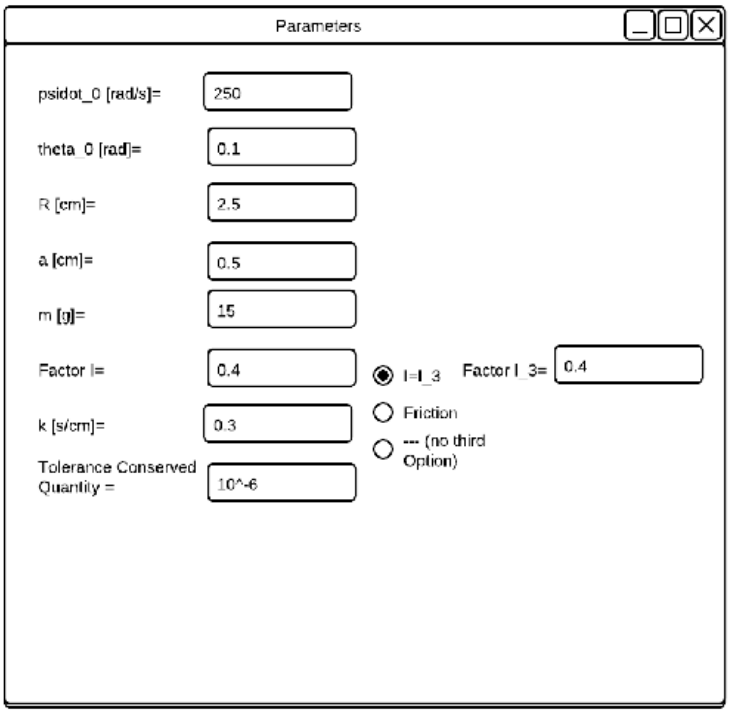
[3] Hairer, E., Nørsett, S.P., and Wanner, G. 1993, Solving Ordinary Differential Equations I. Nonstiff Problems, 2nd ed. (New York: Springer). Fortran codes at http://www.unige.ch/ hairer/software.html

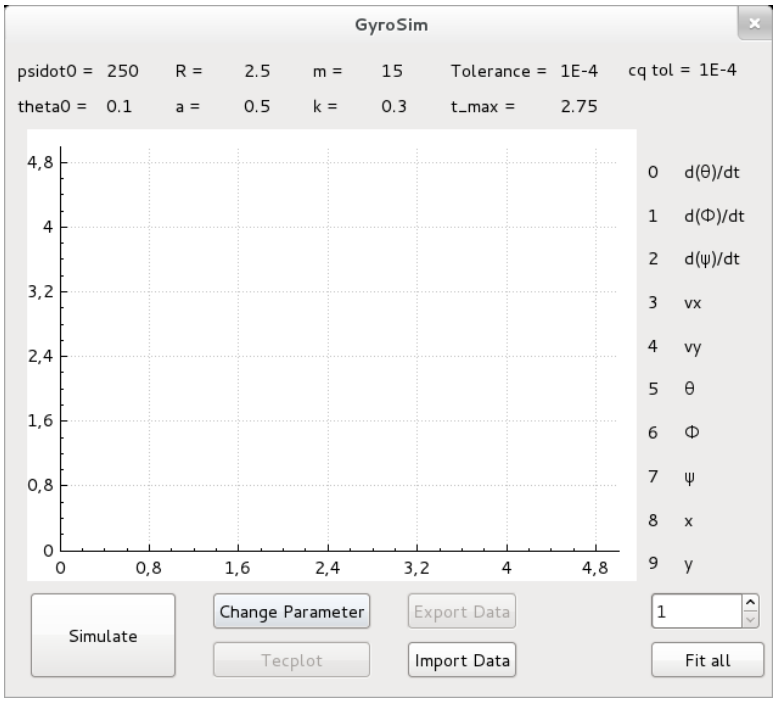
$IR\dot{\phi}\sin^2\theta + I_3(R\cos\theta - a)(\dot{\phi}\cos\theta + \dot{\psi}) = const =: G$
berücksichtigt. (ΔG pro Schritt $< 1E-6$). Der Löser unterstützt sowohl eine dichte Ausgabe *dense output*, als auch die Ausgabe von n äquidistant verteilten Werten.[4]

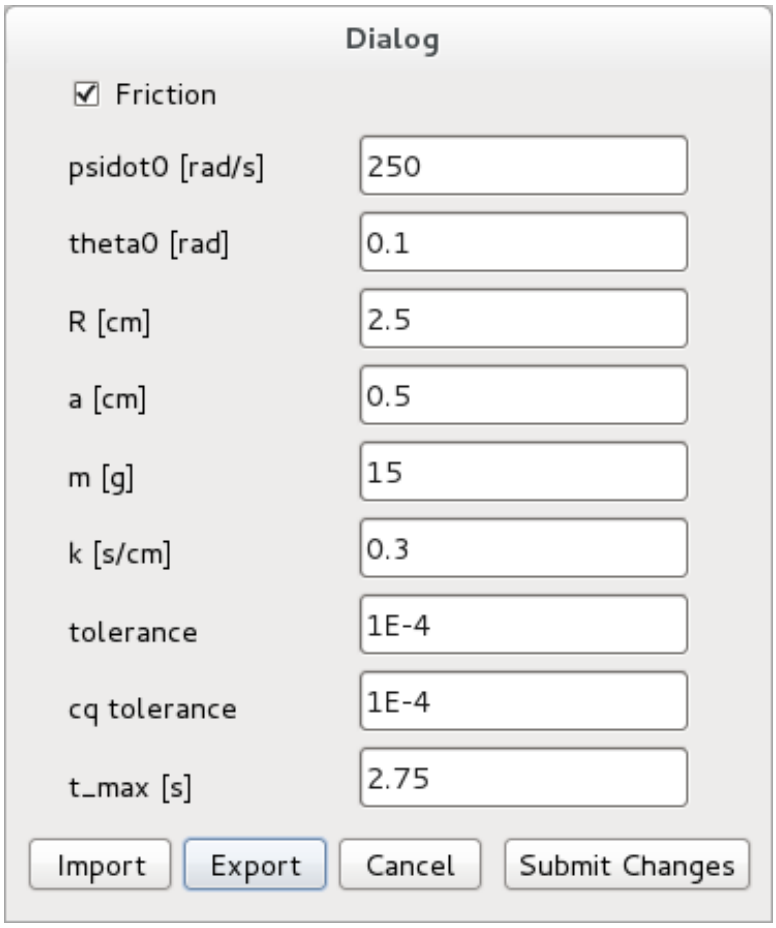
[4]Frei nach NumericalRecipes3rdEdition - Chapter 17.2.4 Dopr853 - An Eight-Order Method Implementierung nach Numerical Recipes Software 2007, "Routine Implementing an Eighth-order Runge-Kutta Method,"Numerical Recipes Webnote No. 20, at http://www.nr.com/webnotes?20

3.3 Graphical User Interface

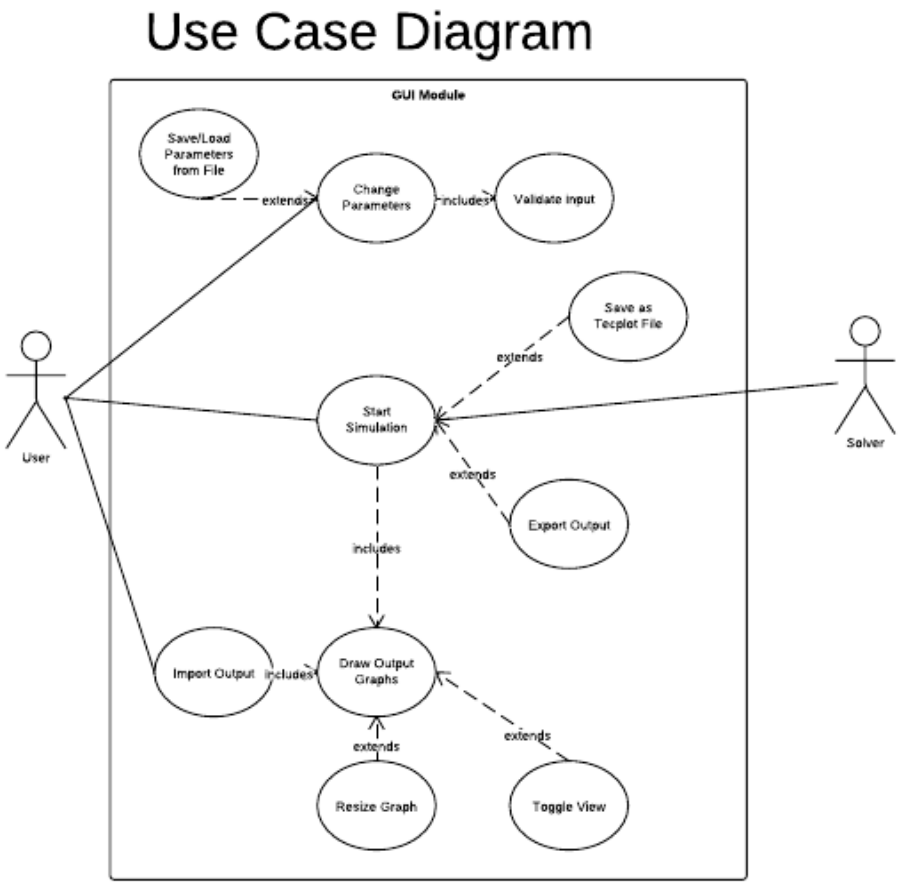





GyroSim
psidot0 = 250
R = 2.5
m = 15
Tolerance = 1E-4
cq tol = 1E-4
theta0 = 0.1
a = 0.5
k = 0.3
t_max = 2.75
4,8
4
3,2
2,4
1,6
0,8
0
0
0,8
1,6
2,4
3,2
4
4,8
0 d(θ)/dt
1 d(Φ)/dt
2 d(ψ)/dt
3 vx
4 vy
5 θ
6 Φ
7 ψ
8 x
9 y
Simulate
Change Parameter
Export Data
1
Tecplot
Import Data
Fit all


Dialog
Friction
psidot0 [rad/s]
250
theta0 [rad]
0.1
R [cm]
2.5
a [cm]
0.5
m [g]
15
k [s/cm]
0.3
tolerance
1E-4
cq tolerance
1E-4
t_max [s]
2.75
Import
Export
Cancel
Submit Changes

3.4 Use-Case-Diagramm

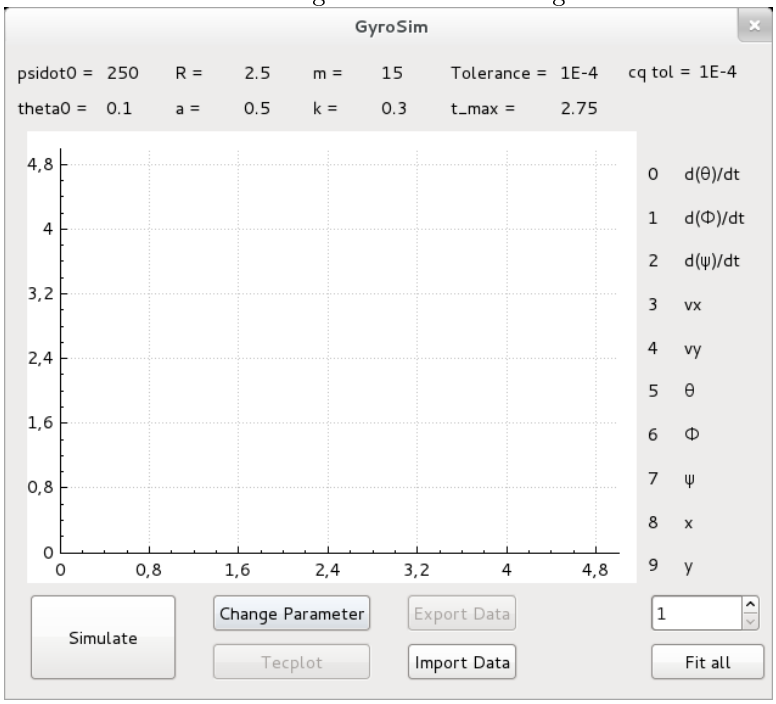


Kapitel 4

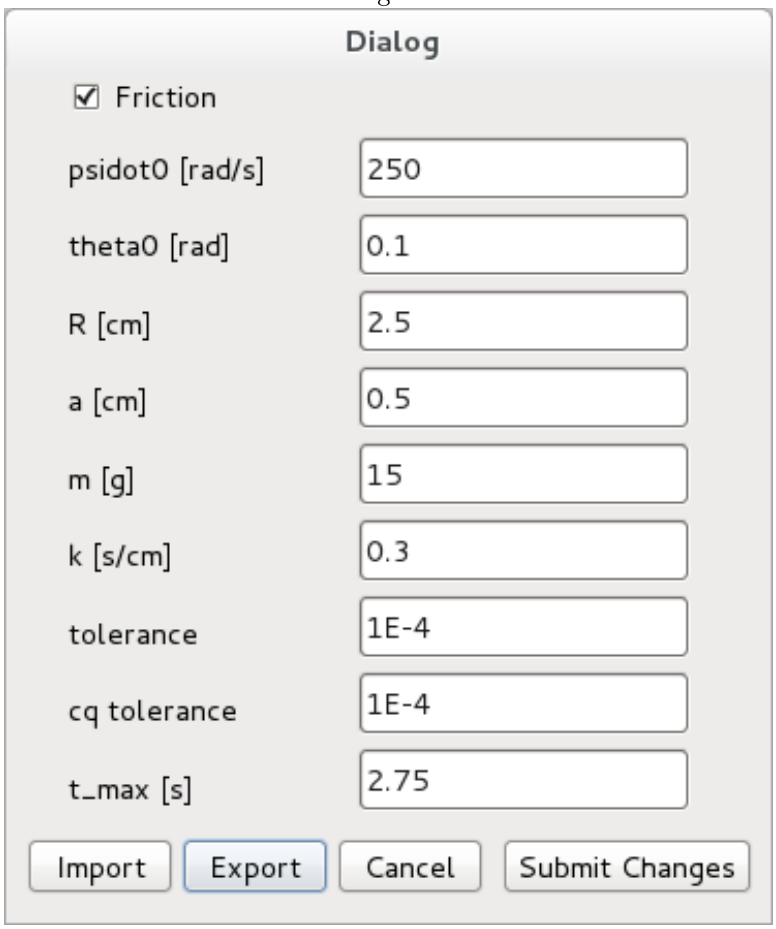
Benutzerdokumentation

4.1 Hauptprogramm

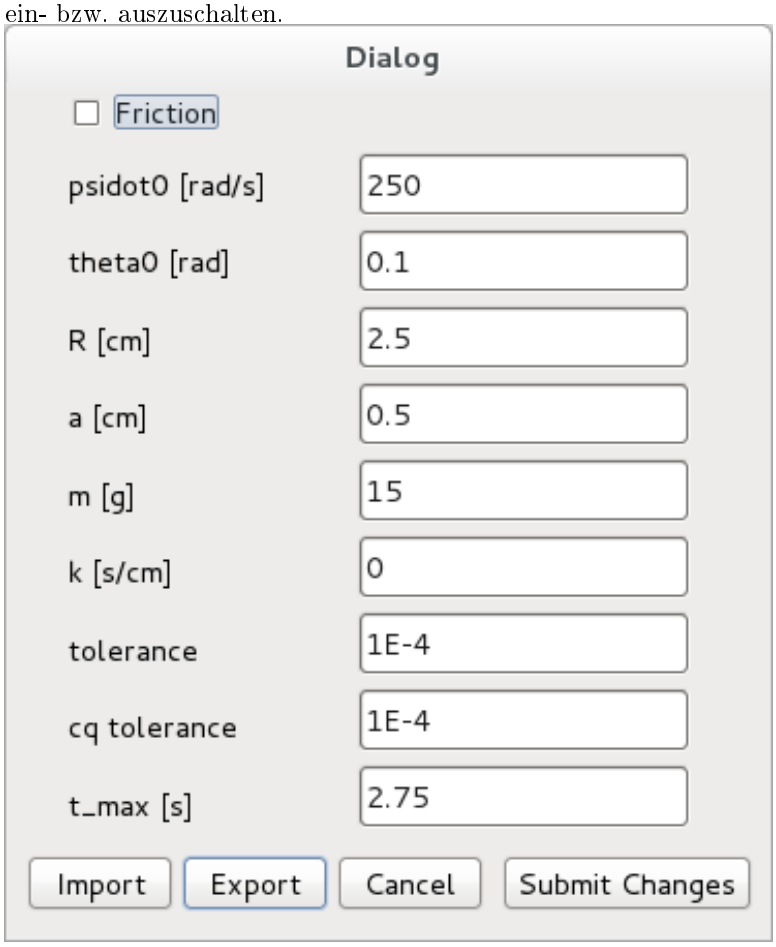
Nach dem Starten des Programms erscheint folgende GUI:



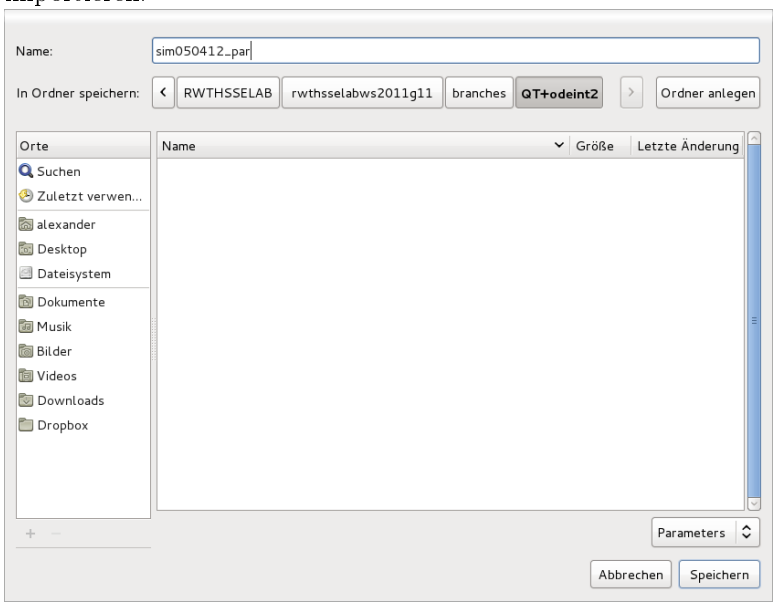
Mit einem Mouseclick auf 'Change Parameter' öffnet sich das Parameterfenster:



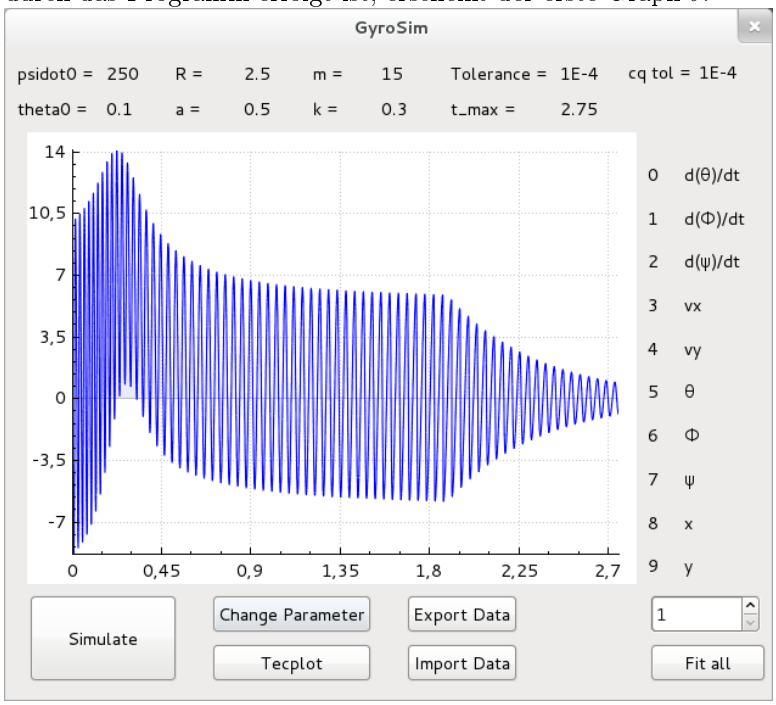
Hier hat der Nutzer die Möglichkeit, die Reibung durch Klicken einer Checkbox ein- bzw. auszuschalten.



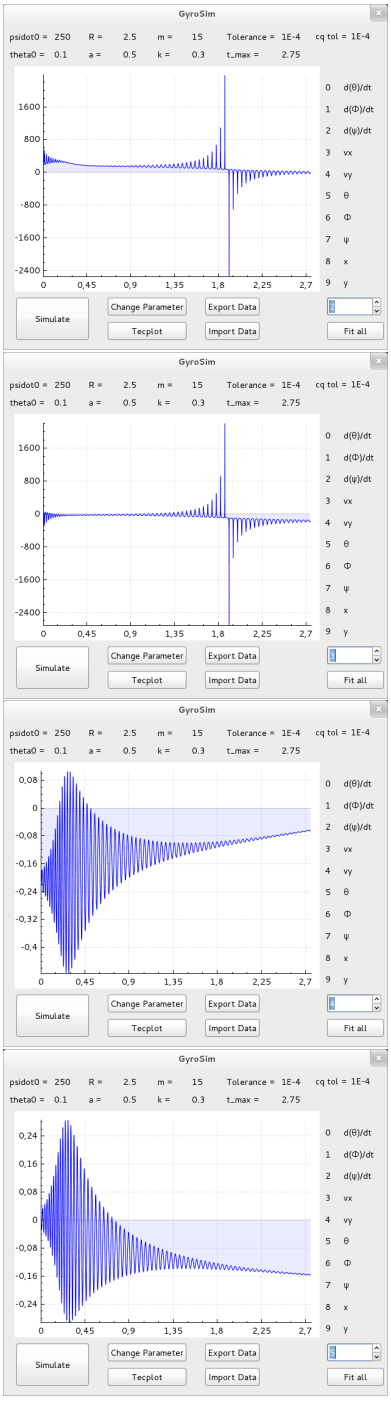
Ebenso kann er eine Parameterkonfiguration als $*.par$ exportieren und wieder importieren.

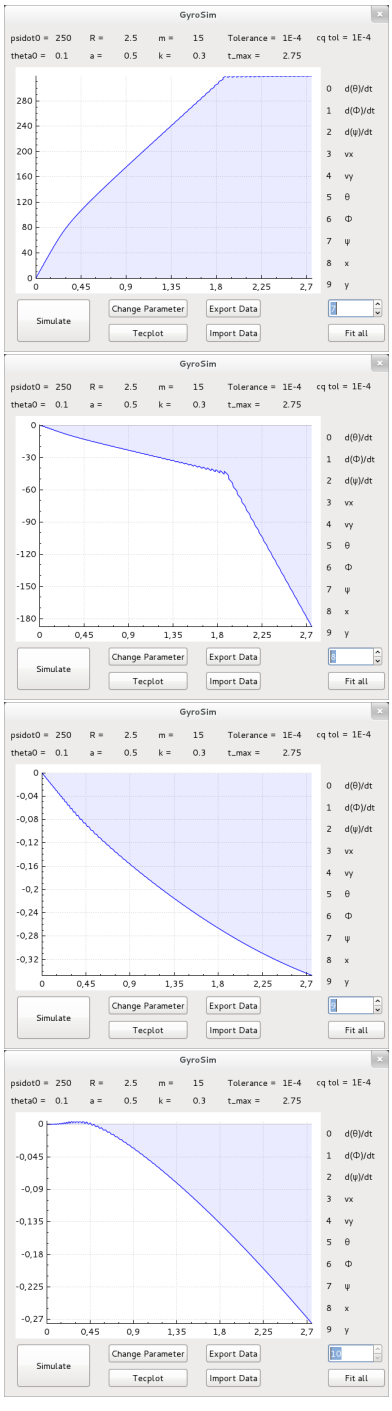


Nachdem die Parameter mit dem Button 'Submit Changes' gesetzt worden sind, klickt der Nutzer im Hauptfenster auf 'Simulate' und nachdem die Berechnung durch das Programm erfolgt ist, erscheint der erste Graph $\dot{\theta}$.

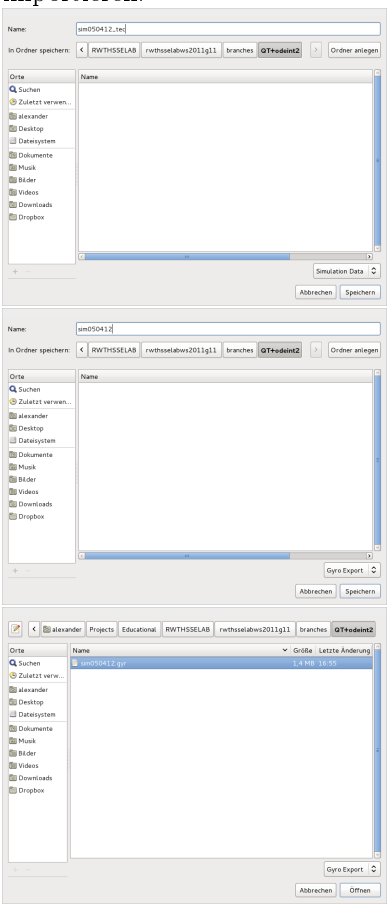


Der Nutzer kann sich auch alle anderen Graphen anzeigen lassen:



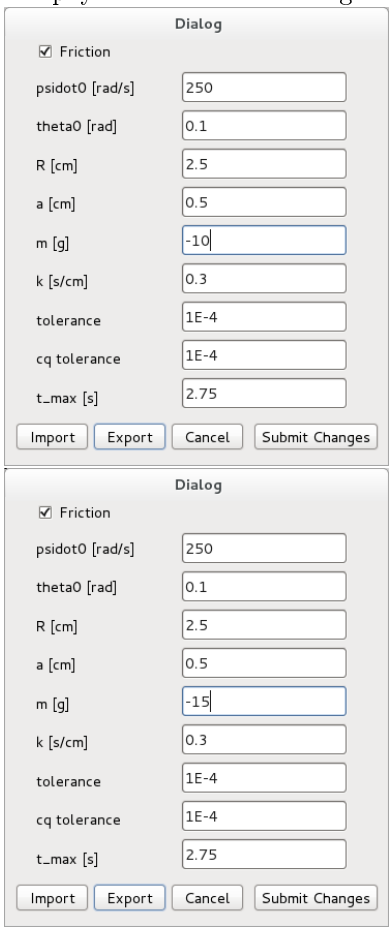

GyroSim
psidot0 = 250 R = 2.5 m = 15 Tolerance = 1E-4 cq tol = 1E-4
theta0 = 0.1 a = 0.5 k = 0.3 t_max = 2.75
0 d(θ)/dt
1 d(Φ)/dt
2 d(ψ)/dt
3 vx
4 vy
5 θ
6 Φ
7 ψ
8 x
9 y
Simulate
Change Parameter
Export Data
Tecplot
Import Data
Fit all

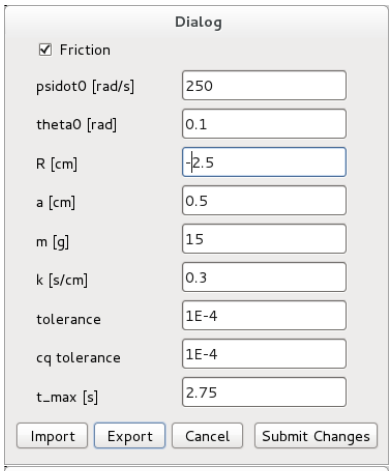
Im Hauptfenster kann der Nutzer nach erfolgter Simulation die Daten entweder als Tecplot oder als $*.gyro$ speichern und bereits exportierte Daten wieder importieren.

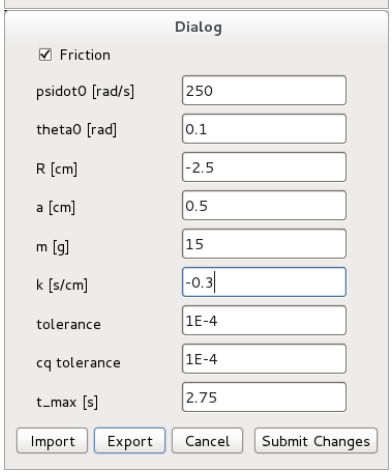


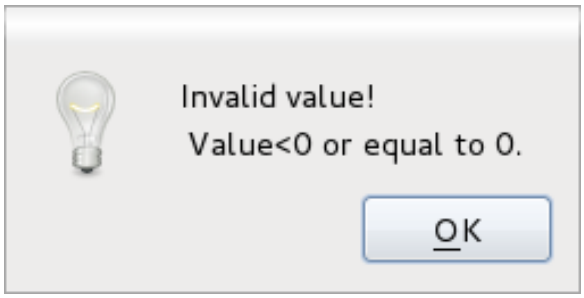
4.2 Fehlermeldungen

Bei physikalisch falschen Eingaben erfolgt eine Fehlermeldung.




Dialog
Friction
psidot0 [rad/s]
250
theta0 [rad]
0.1
R [cm]
-2.5
a [cm]
0.5
m [g]
15
k [s/cm]
0.3
tolerance
1E-4
cq tolerance
1E-4
t_max [s]
2.75
Import
Export
Cancel
Submit Changes


Dialog
Friction
psidot0 [rad/s]
250
theta0 [rad]
0.1
R [cm]
-2.5
a [cm]
0.5
m [g]
15
k [s/cm]
-0.3
tolerance
1E-4
cq tolerance
1E-4
t_max [s]
2.75
Import
Export
Cancel
Submit Changes


Invalid value!
Value<0 or equal to 0.
OK

Kapitel 5

Entwicklerdokumentation

Im separaten Dokument *Entwicklerdokumentation* zu finden.