**Simulation Fallschirmspringer**

Patrice Keusch, Hauptrasse 30, Döttingen 5312, Tel. 056 633 99 63, [keuscpat@students.zhaw.ch](mailto:keuscpat@students.zhaw.ch)

Severin Müller, Kesselstrasse 20, 8200 Schaffhausen, Tel. 052 620 40 09

Softwareprojekt 2 mit Anwendungen von Methoden aus dem Bereich der Nummerischen Mathematik

ZHAW

Kursverantwortlicher: Philippe Nahlik

Scrummaster: Lukas Eppler

Auftraggeber: Albert Heuberger

12.06.2012

Inhaltsverzeichnis

[1 Einleitung 4](#_Toc327107607)

[1.1 Ausgangslage 4](#_Toc327107608)

[1.2 Aufgabenstellung 4](#_Toc327107609)

[1.3 Eigenmotivation 4](#_Toc327107610)

[1.4 Vorgehen 5](#_Toc327107611)

[1.5 Teilziel 1 – Mathematische Grundlagen und GUI 5](#_Toc327107612)

[1.6 Teilziel 2 6](#_Toc327107613)

[1.7 Teilziel 3 6](#_Toc327107614)

[1.8 Teilziel 4 6](#_Toc327107615)

[2 Analyse 7](#_Toc327107616)

[2.1 Luftwiderstand 7](#_Toc327107617)

[2.2 Bewegungsgesetz von Newton 7](#_Toc327107618)

[2.3 Differentialgleichung 7](#_Toc327107619)

[2.4 Zustand des Springers 8](#_Toc327107620)

[2.5 Widerstand des Windes 8](#_Toc327107621)

[2.6 Ablauf der Simulation 9](#_Toc327107622)

[2.6.1 Phase 1 – Flugzeug 9](#_Toc327107623)

[2.6.2 Phase 2 – Absprung aus Flugzeug 10](#_Toc327107624)

[2.6.3 Phase 3 – Beschleunigung bis maximale Geschwindigkeit 10](#_Toc327107625)

[2.6.4 Phase 4 – Fallschirm wird geöffnet 10](#_Toc327107626)

[2.6.5 Phase 5 – Fallschirmspringer gleitet 11](#_Toc327107627)

[2.6.6 Phase 6 – Landen am Zielpunkt 11](#_Toc327107628)

[2.7 Runge-Kutta Verfahren 11](#_Toc327107629)

[2.8 Vektorrechnungen 12](#_Toc327107630)

[2.9 Exakte Lösungen finden 12](#_Toc327107631)

[2.10 Grafische Darstellung 12](#_Toc327107632)

[3 Umsetzung 13](#_Toc327107633)

[3.1 SimulationObject 13](#_Toc327107634)

[3.2 Vektor Berechnungen 13](#_Toc327107635)

[3.3 Windfunktion 13](#_Toc327107636)

[3.4 Widerstandfunktion 13](#_Toc327107637)

[3.5 Springer 14](#_Toc327107638)

[3.5.1 Schrittweite Berechnung 14](#_Toc327107639)

[3.5.2 Springer Flugbahn berechnen 14](#_Toc327107640)

[3.6 Runge-Kutta Funktion 15](#_Toc327107641)

[3.7 Berechnungen der Ableitungen 16](#_Toc327107642)

[3.8 Exakte Lösungen finden 17](#_Toc327107643)

[3.9 Grafische Darstellung 17](#_Toc327107644)

[4 Software (SM) 18](#_Toc327107645)

[4.1 Grundstruktur 18](#_Toc327107646)

[4.2 Klassendiagram 18](#_Toc327107647)

[5 Simulation - Kurze Flugzeit 19](#_Toc327107648)

[6 Fazit 22](#_Toc327107649)

[6.1 Sevi 22](#_Toc327107650)

[6.2 Patrice 22](#_Toc327107651)

[6.3 Danksagung 22](#_Toc327107652)

[7 Anhang 23](#_Toc327107653)

[7.1 Quellenverzeichnis 23](#_Toc327107654)

[7.2 Abbildungsverzeichnis 23](#_Toc327107655)

# Einleitung

## Ausgangslage

Dieses Projekt wurde im Kurs „Softwareprojekt 2“ im vierten Semester des Studiengangs Informatik am Standort Zürich der ZHAW realisiert. Ziel des Kurses war, ein kleines Projekt zu realisieren, welches professionelle Standards erfüllt und im Inhalt eine Anwendung von Methoden aus den Bereichen Numerische Mathematik, Algorithmen und Datenstrukturen oder Theoretische Informatik sein soll. Die Aufgabenstellung und die fachliche Betreuung übernimmt ein Dozent in der Rolle des Auftraggebers. Ein weiterer Dozent übernimmt die Rolle des Scrummasters, welcher die Projektplanung und Projektumsetzung überwacht.

## Aufgabenstellung

Als Aufgabenstellung haben wir uns die Simulation eines Fallschirmsprunges ausgesucht. Für eine realitätsnahe Simulation braucht es numerische Methoden, Differentialgleichungen und Physikalische Gleichungen.

Wir überlegen uns folgendes Szenario: Ein Fallschirmspringer bereits im Flugzeug und macht sich bereit für seinen Absprung. Diverse Parameter wie Windstärke, Gravitation, Flugzeuggeschwindigkeit usw. sind bekannt. Der Fallschirmspringer möchte nun einen bestimmten Landepunkt erreichen. Er kann die Koordinaten des gewünschten Zieles im Programm eingeben und seine persönnlichen Parameter ergänzen. (Gewicht, Fallschirmgrösse, nach welcher Zeit der Fallschirm geöffnet wird etc.). Das Programm soll nun anhand der physikalischen Gleichungen und numerischen Verfahren zum Lösen von Differentialgleichungen den Abspringpunkt aus dem Flugzeug so berechnen, dass der Springer eine Punktlandung hinlegt. Mit Hilfe dieses Programmes können dann verschiedene Zeiten für das Öffnen des Fallschirmes berechnet werden. Je nach dem was gewünscht ist, kann so die längste oder kürzeste Flugzeit berechnet werden, ohne dass sich der Springer bei der Landung verletzt.

## Eigenmotivation

Bei der Themenwahl konnten wir uns mit keinem der vorgegebenen Themen so richtig anfreunden. Wir überlegten uns daher eine eigene Aufgabenstellung. Die Idee für die Simulation eines Fallschirmsprunges kam uns durch Spiel, wo man einen Fallschirmspringer fallen lässt und dann im richtigen Augenblick den Fallschirm öffnen muss, so dass dieser auf der Zielplatform landet und auch nicht zu schnell beim Aufprall ist.

Zudem ist ein Freund von Patrice Keusch ein „Sky Diver“, was das Thema umso interessanter machte und man auch von realen Erfahrungen profitieren konnte.

Die Vorstellung einen natürlichen Vorgang wie der Fallschirmsprung mit Hilfe einer Differentialgleichung zu beschreiben und dies mit nummerischen Verfahren zu lösen, motivierte uns sehr. Wir haben uns bisher viel theoretisches nummerisches Wissen angeeignet und waren nun gespannt darauf, wie sich dies bei einer realen Simulation verhält.

Schlussendlich hat das Software Projekt mit der Umsetzung der Simulation in einer Programmiersprache zu tun. In der Berufstätigkeit von Patrice Keusch kommt das Programmieren jedoch zu kur, was es interessant macht, sich intensiv mit der Programmierung zu beschäftigen. Severin Müller arbeitet zwar als Entwickler, jedoch nicht in einem Java Umfeld, was es für ihn ebenfalls spannend macht.

Unser Ziel war ganz klar, die Aufgabenstellung möglichst gut zu erfüllen und die berechneten Werte in einem GUI aussagekräftig darzustellen. Dafür mussten wir uns mit der Programmierung eines GUI ausseinander setzen und mit der Umsetzung von numerischen Methoden in einer Hochsprache.

## Vorgehen

Für die Umsetzung des Projekts haben wir uns für ein paar Hilfsmittel entschieden:

* Iterationsplan mit User Stores und Tasks
* Dynamischer Iterationsplan (Google Docs)
* Dynamisches Fortschrittsdokument (Google Docs)

Als erstes haben wir zusammen mit dem Kunden die Aufgabenstellung festelegt. Danach haben wir uns mit dem Thema auseinandergesetzt, um dann einen fixen Iterationsplan zu erstellen. Im Iterationsplan haben wir uns User Stories und die passenden Tasks dazu überlegt. Wir haben uns dabei auch Gedanken über die zur Verfügung stehende Zeit, Velocity und Task-Aufteilung Gedanken gemacht. Für das Projekt müssen wir pro Person ca 60 Stunden aufwenden und wir haben ca 11 Wochen Zeit. Wenn wir nun die Velocity einberechnen und die Anzahl Wochen beachten, gibt dies ca 8 Stunden pro Woche, welche wir investieren können.

Um das Projektmanagement dynamisch zu halten, haben wir uns entschieden einen Iterationsplan und ein Fortschrittsdokument auf Google Docs zu erstellen. Im Iterationsplan ist jeweils der aktuelle Status der Tasks und User Stories ersichtlich. Im Fortschrittsdokument werden Meeting Ergebnisse, Tätigkeiten und ToDo-Listen abgelegt.

Bei der Erarbeitung des Iterationsplans haben wir uns für vier Teilziele entschieden. Wir wollten dabei vorallem das Risiko vermeiden, dass wir zuletzt noch vor Grossen Problemstellungen stehen und nicht rechtzeitig fertig werden.

## Teilziel 1 – Mathematische Grundlagen und GUI

Befor wir mit der Programmierung beginnen, mussten die mathematischen Grundlagen und die physikalischen Gleichungen klar sein. Der Ablauf eines Fallschirmsprunges muss verstanden sein und Überlegungen zur Lösung mit Hilfe von nummerischen Methoden muss erarbeitet werden.

Das Grundlayout des grafischen User Interfaces muss stehen und es muss klar sein, wie wir die Flugbahn grafisch darstellen können. Hochsprache und Frameworks müssen festgelegt sein.

## Teilziel 2

Programmierung der nummerischen Methoden und erste Berechnungen müssen möglich sein. Einbeziehen der korrekten Physikalischen Gleichungen. (Widerstand, Windstärke etc.)

Die mathematischen Berechnungen und das GUI müssen zusammen agieren und es können erste Koordinaten gezeichnet werden im GUI. Im GUI können die Paramter definiert werden und es finden Validierungen statt.

## Teilziel 3

Die Software wird komplettiert und verfeinert. Das GUI wird mit nützlichen Funktionen und Darstellungen erweitert. Die Flugbahn kann berechnet und dargestellt werden mit diversen Parametern. Die Lösung der Aufgabenstellung ist erreicht.

## Teilziel 4

Die Konzeptdokumentation ist fertiggestellt. Der Programmcode ist bereinigt und dokumentiert. Die Abschlusspräsentation ist vorbereitet und einstudiert. Die CD für die Abgabe wird erstellt mit folgenden Inhalten:

* Handout der Präsentation
* Code, Dokumentiert
* Allenfalls notwendige Libraries zum Ausführen des Programms
* Konzeptdokument, welches die Anforderungen enthält
* Projektplan, allenfalls im Konzeptdokument vorhanden
* In einer üblichen Umgebung lauffähiges Programm

# Analyse

In der Analyse wird die Problemstellung analysiert und einzelne Themen genauer angeschaut. Benötigte physikalische Gleichungen und nummerische Verfahren werden erklärt. Entsprechende Funktionen werden gleich definiert.

## Luftwiderstand

Der Luftwiderstand wird von vier Faktoren beeinflusst:

1. Faktor: Der Luftwiderstand ist proportianal zu der grössten Querschnittsfläche A, die senkrecht zur Strömungsrichtung steht
2. Faktor: Der Luftwiderstand ist proportional zur Geschwindigkeit v im Quadrat.
3. Faktor: Der Luftwiderstand ist proportional zur Luftdichte .
4. Faktor: der Luftwiderstand hängt von der Form des Körpers ab.

Der Luftwiderstand kann daher durch diese Formel beschrieben werden:

Der Faktor cw bezeichnet den sogenannten Formfaktor. Je nach Form des Objektes, ist dieser Wert höher oder kleiner. Dieser Wert muss normalerweise in echten Simulationen gemessen werden. In unserer Simulation starten wir mit einem Wert aus Dokumentationen für den freien Fall. Den cw Wert während des Öffnen des Fallschirmes und des offenen Fallschirmes werden wir mit Programnsimulationen ermitteln.

## Bewegungsgesetz von Newton

Für die Simulation brauchen wir das Bewegungsgesetzt von Newton. Es wirken diese Kräfte auf den Fallschirmspringer:

Schwerkraft:

Widerstandskraft:

Gesamtkraft:

## Differentialgleichung

Wir wissen nun, dass auf den Körper zwei Kräfte wirken: Die Gewichtskraft und die Widerstandskraft, die in die entgegengesetzte Richtung gerichtet ist.

Die Gesamtkraft können wir nun beschreiben.

Daraus können wir die Bewegungsgleichung aufstellen.

Formen wir diese Differentialgleichung um, erhalten wir eine passende Form für zukünftige Berechnungen, mit welcher wir die Beschleunigung berechnen können.

Mit Hilfe dieser Differentialgleichung und mit dem RK-Verfahren können wir die jeweilige Beschleunigung zu der Zeit t berechnen und können dadurch die Flugbahn ermitteln.

## Zustand des Springers

Der Springer ist das zentrale Objekt in unserer Berechnung. Er muss zu jedem Zeitpunkt seine Position und seine Geschwindigkeit kennen.

Der Zustand des Springers wird durch einen Vektor z in Abhängikeit zu t beschrieben:

Die Positionsvektoren abgeleitet ergibt die Geschwindigkeitsvektoren und die Geschwindigkeitsvektoren abgeleitet ergibt die Beschleunigungsvektoren.

Mit der aufgestellten Differentialgleichung können wir demnach die benötigten Werte berechnen.

## Widerstand des Windes

Der Wind ist ein weitere Faktor aus der Realität. Der Wind wird als Windwidertand betrachtet.

Die Windstärke ist definiert:

Je nach Positionshöhe ist die Windstärke verschieden. Die Windfunktion W(y) ist abhängig von der y Koordinate. Um die Simulation interessant zu gestalten, haben wir verschiedene Werte angenommen.

Um nun die Windfunktion einzubeziehen, wird diese jeweils zu dir aktuellen Geschwindigkeit addiert, befor danach die neue Geschwindigkeit berechnet wird:

## Ablauf der Simulation

Wir beschreiben den Ablauf der Simulation sehr detailliert. Bevor mit Berechnunungen begonnen wird, muss der Ablauf verstanden sein.

### Phase 1 – Flugzeug

Zuerst ist der Fallschirmspringer im Flugzeug. Der Fallschirmspringer besitzt daher die Beschleunigung *a*, Geschwindigkeit *v* und die Koordinaten des Flugzeugs.

Für die Simulation der Flugbahn des Flugzeuges können wir eine einfache Funktion verwenden. Wir vernachlässigen hier den Luftwiderstand, Gravitation und weitere Einflussfaktoren. Wir gehen davon aus, dass wir das Flugzeug immer mit der gleichen Geschwindigkeit bewegen und in der gleichen Höhe halten.

Es ergibt sich daraus folgende Ortsfunktion:

Wobei v2 = 0 ist in unserer Simulation und v1 beliegbig gesetzt werden kann.

### Phase 2 – Absprung aus Flugzeug

Beim Absprung übernimmt der Springer die Werte des Flugzeuges zum Zeitpunkt t:

Der Springer wird danach immer schneller, zudem passt sich die v1 Geschwindigkeit der Windgeschwindigkeit in dieser Höhe an.

### Phase 3 – Beschleunigung bis maximale Geschwindigkeit

Die Springer erreicht seine maximale Geschwindigkeit nach ca 10 Sekunden, abhängig von Gewicht und der vertikalen Windstärke. Die maximale Geschwindigkeit kann berechnet werden, in dem man die Beschleunigung auf 0 setzt, daher keine Beschleunigung mehr vorhanden ist:

Durch umformen erhält man die Endgeschwindigkeit:

Je grösser die Fläche, desto kleiner wird die Endgeschwindigkeit. In unserer Simulation legen wir die Startfläche des Springers auf . In der Realität ändert sich diese Fläche immer wieder, je nach Haltung des Springers. Durch eine möglichst senkrechte Haltung würde der Springer die höchste Geschwindigkeit erreichen und durch eine waagrechte Haltung mit gestreckten Armen eine relativ tiefe Endgeschwindigkeit.

### Phase 4 – Fallschirm wird geöffnet

In unserer Simulation können wir den Zeitpunkt des Öffnens vorgeben. Der Fallschirm braucht für das Öffnen des Fallschirms ca 2-5 Sekunden. In unserer Simulation haben wir den Wert auf 2 Sekunden gesetzt. In dieser Zeit ändert sich die Fläche von bis auf die eingestellt Fallschirm Fläche zum Beispiel 20.

Ebenso ändert sich der cw-Wert in diesen zwei Sekunden. In unserer Simulation ist der cw Start Wert bei 0,5 und der Endwert bei 3,0.

Die Widerstandsfunktion wird daher erweitert:

Bei jedem Berechungsschritt wird daher die Funktion A(t) und cw(t) neu berechnet. Abhängig der Schrittweite der Berechnung, wird A und cw Wert linear erhöht, bis der Fallschirm offen ist.

### Phase 5 – Fallschirmspringer gleitet

Da der Widerstand durch den offenen Fallschirm sehr gross ist, gleitet der Fallschirmspringer relativ langsam hinunter. Die xEndgeschwindigkeit darf nun nicht mehr sehr hoch sein, da der Springer mit dieser Geschwindigkeit auf dem Boden landen wird. Beim Fallschirmspringen wird die Grösse das Fallschirms so gewählt, das die Endgeschwindigkeit bei der Landung zwischen 3,5 und 5,0m/s liegt. Was ca 15km/h entspricht.

Wie in allen Phasen nährt sich die Geschwindigkeit in xRichtung der Geschwindigkeit des Windes zur gegeben Höhe annehmen an, bis diese erreicht wird.

### Phase 6 – Landen am Zielpunkt

Sobald die Koordinate y = 0 ist. Ist der Springer gelandet. Dabei kommt es abrupt zur Veringerung der Geschwindigkeit auf 0.

## Runge-Kutta Verfahren

Als nummerisches Verfahren zur Lösung der Differentialgleichung haben wir uns für das Runge-Kutta Verfahren entschieden. Für das Verständnis und die ersten Versuche haben wir ebenfalls das Euler-Verfahren verwendet. Das Euler-Verfahren ist einfacher anzuwenden, hat jedoch nur eine Konsistenz – und Konvergenzordnung von p=1, wogegen das RK-Verfahren p=4 besitzt. Daher werden wir das RK-Verfahren einsetzen, welches die wesentlich besseren Näherungen berechnet.

Wie der Euler und das RK-Verfahren in Java eingesetzt werden kann, haben wir bei Herrn Heuberger in einer Schulung gelernt und dafür auch den entsprechenden Java Code erhalten. Wir haben den Code für unsere Berechnungen angepasst und erweitert.

Das Runge-Kutta Verfahren haben alle Studenten im Kurs „Numerik 2“ kennen gelernt, daher wird der Algorithmus nicht näher erklärt.

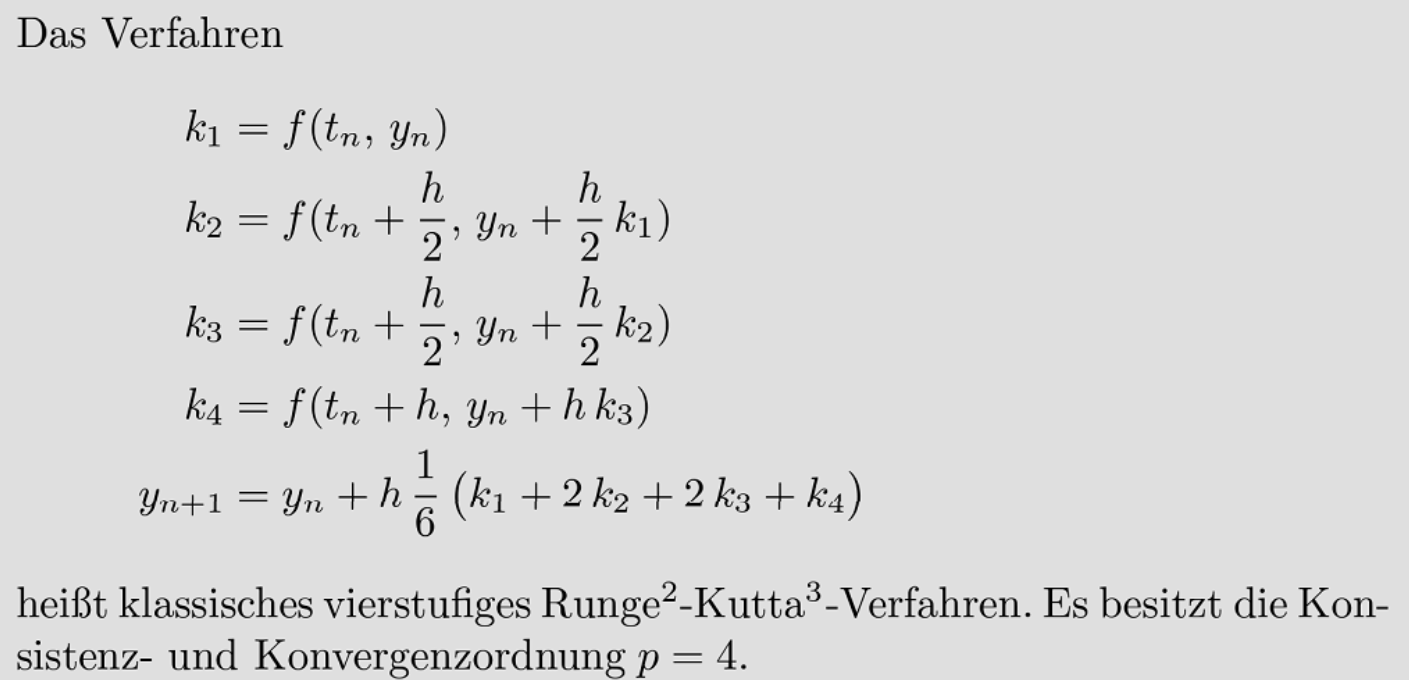


Abbildung Seite 150 Nummerische Mathematik

## Vektorrechnungen

Da wir im zweidimensionalen Raum arbeiten, brauchen wir diverse Vektorrechnungen.

AddVektor, MultiVektor, meanVektor

## Exakte Lösungen finden

Um die exakte Lösung zu finden, können wir eine simple Überlegung anstellen. Wir berechnen die Werte mit dem RK-Verfahren und geben als Startwert eine mögliche Koordinate ein wie zum Beispiel (1000,0). Diese Berechnung wird uns einen Landepunkt ausgeben, wo die yKoordinate = 0 ist. Nun haben wir einen berechneten Wert x. Wollten wir nun bei der Koordinate 300 landen, sind aber auf der Koordinate 500 gelandet, können wir das RK-Verfahren nochmals mit geändertem Startwert berechnet lassen. Wir würden die Berechnungen in diesem Beispiel mit den Werten (1000,200) starten. Da die Berechnung symetrisch ist und die Parameter sich nicht mehr ändern, landet der Fallschirmspringer bei der zweiten Berechnung exakt am gewünschten Zielort.

## Grafische Darstellung

# Umsetzung

## SimulationObject

Die Klasse SimulationObject.java verwenden wir als DatenObjekt. Darin sind über 25 Variablen deklariert, welche mit Getter Methoden abgerufen und mit Setter Mehtoden gesetzt werden können.

## Vektor Berechnungen

Für die Vektor Berechnungen verwenden wir die Klasse linalg4\_4.java. Diese Klasse hat uns Herr Heuberger zur Verfügung gestellt. Die Klasse Springer.java erbt die Methoden der Klasse linalg4\_4.java, wodurch wir die benötigten Vektor Berechnungsmethoden aufrufen können.

## Windfunktion

Zu jedem Zeitpunkt setzen wir die Windstärke neu. Je nach Höhe wird ein anderer Wind festgelegt.

|  |
| --- |
| Ein Teil der *wind()* Funktion aus Springer.java |
|  |

Die Windgeschwindigkeit wird dann bei der Berechnungen der Ableitungen abgezogen. Wobei das Array die neuen Beschleunigungsvektoren speichert.

|  |
| --- |
| Ein Teil der w Funktion aus Springer.java |
|  |

## Widerstandfunktion

Im Gegensatz nur Analyse verwenden wir hier für den Widerstand vorerst nur diese Gleichung:

Da die Geschwindigkeit durch Vektoren definiert ist, lassen wir hier den Faktor noch weg, werden diesen aber in den Berechnungen der Ableitungen nachträglich mit einbeziehen.

Sobald der Fallschirm geöffnet wird, wird der cw und der Flächenwert schrittweise erhöht.

|  |
| --- |
| *calcWiderstand()* und *calcCW()* Funktion aus Springer.java |
|  |

## Springer

Die Springer Klasse enthält sämtliche Berechnungen. Bei der Instanzierung einer Springer Klasse muss ein SimulationObject mitgegeben werden. Die Klasse Springer wird dann mit allen wichtigen Werten initialisiert.

### Schrittweite Berechnung

Das Runge-Kutta Verfahren kann mit verschiedenen Schrittweiten berechnet werden, daher muss die Schrittweite für die Fallschirmfläche und den cw-Wert während des Öffnens vor jeder Berechnung nue ermittelt werden. Dies wird bei der Initialisierung ausgelöst.

|  |
| --- |
| CalcSchritte Funktion aus Springer.java |
|  |

### Springer Flugbahn berechnen

Die Flugbahnberechnung wird mit der Methode *calcSpringer()* gestartet.

Die Funktion *fTable()* startet das Runge-Kutta Verfahren und gibt danach ein 5-Array aus. Folgende Parameter werden dort mitgegeben:

|  |  |
| --- | --- |
| Ausgabe tStart | Ab welchem t soll die Tabelle ausgegeben werden |
| Ausgabe Schrittweite | Welche Schrittweite für die Ausgabe |
| Ausgabe tEnde | Bis welche Laufzeit soll die Ausgabe erstell werden |
| tStart | Startzeit der Berechnungen |
| yAnfang | Werte für das Anfangswertproblem |
| yAnfang[0] | X Koordinate |
| yAnfang[1] | Y Koordinate |
| yAnfang[2] | Flugzeuggeschwindigkeit |
| yAnfang[3] | Springer Geschwindigkeit |
| yAnfang[4] | Variable für Laufzeit |
| Schrittweite | h Schrittweite für Runge-Kutta |

Durch die zwei verschiedenen Schrittweiten kann erreicht werden, dass mit einer kleiner Schrittweite gerechnet wird, die Ausgabe jedoch eine grössere Schrittweite aufweist. Dadurch bekommt man genaue Resultate, jedoch muss pro Sekunde nicht 1000-Mal das gleiche Objekt auf fast den gleichen Punkt gezeichnet werden.

|  |
| --- |
| Auszug von CalcSpringer Funktion aus Springer.java |
|  |

## Runge-Kutta Funktion

Die Runge-Kutta Funktion wird aus der *fTable()* Funktion ausgerufen und bekommt die Anfangswertprobleme mitgeliefert.

Dies ist ein Auszug aus einer Iteration im RK-Verfahren. Als erstes wird der aktuelle Widerstand mit der Funktion *calcWiderstand()* berechnet. Danach wird gemäss RK-Algorithmus vorgegangen und die K1-4 berechnet. Dort wird jeweils die Funktion *w()* aufgerufen mit der Laufzeit und dem Array mit den Anfangswertsproblemen. Die Funktion *w()* übergibt die neu berechneten Vektoren. Nachdem die Vektoren mit der Schrittweite multipliziert wurden, werden die neuen Vektoren zum alten Vektor addiert und man erählt so die neuen Werte.

|  |
| --- |
| Auszug von RK4 Funktion aus Springer.java |
|  |

## Berechnungen der Ableitungen

Die Funktion *w()* ist die komplizierteste Methode in unserer Software. Es beinhaltet die Ableitungen für die vier Zustandsgrössen. Die uns bekannte Differentialgleichung für die Flugbahn im eindimensionalen Fall, mussten wir nun so umschreiben, dass diese auch im zweidimensionalen Fall mit x und y Vektoren funktioniert.

Die Geschwindigkeitsvektoren ergeben die neuen Postionsvektoren und können daher einfach zugewiesen werden:

|  |
| --- |
| Auszug von w Funktion aus Springer.java |
|  |

Wir berechnen den Betrag der Geschwindigkeitsvektoren und speichern diese in . Mit Hilfe des Einheitsvektors, können wir die neuen Vektor berechnen.

Einheitsvektor:

Zudem komplettieren wir den Widerstandsfaktor r und multiplizieren (). Für den Beschleunigungsvektor ax müssen wir keine Gravitation beachten. Daher lassen wir dort die Gravitation weg.

Setzen wir diese Überlegungen in unsere Differentialgleichung, erhalten wir diese Berechnung.

Wobei wir uBetrag wegkürzen können.

|  |
| --- |
| Auszug von *w()* Funktion aus Springer.java |
|  |

## Exakte Lösungen finden

Nach dem ersten Durchlauf der Berechnungen wird der Landepunkt *KoNull* ausgelesen. Wenn dieser nicht mit dem gewünschten Landepunkt übereinstimmt, wird der Springer neu initalisiert und die Berechnung mit geänderten Anfangswerten nochmals gestartet.

|  |
| --- |
| Auszug von calcSpringer Funktion aus Springer.java |
|  |

## Grafische Darstellung

# Software (SM)

## Grundstruktur

SimulationObject als Datenhalde... Objekt wird abgefüllt im GUI, wird zur Berechnung verwendet, wird zur Darstellung weitergereicht... etc...

## Klassendiagram

# Simulation - Kurze Flugzeit

Mit Hilfe der Software können wir den Abspringpunkt und die Zeit des Öffnens durch ausprobieren bestimmen. Dazu haben wir ein paar Versuche gemacht. Das Ziel in dieser Simulation ist es, die kürzeste Flugzeit zu erreichen. Dass heisst, der Fallschirm wird möglichst spät geöffnet. Wichtig ist dabei, dass der Fallschirmspringer bei der Landung keine grosse Geschwindigkeit mehr hat, da er sonst den Aufprall nicht überlebt. In der Realität landen Fallschirmspringer mit , wobei eine bereits eine harte Landung ist. Ab einer Geschwindigkeit höher als muss bereits mit Verletzungen gerechnet werden.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Simulation – Versuch 1** | | | | |
| **Paramter** | |  | **Resultate** | |
| Flughöhe |  |  | Flugzeit |  |
| Flugzeuggeschwindigkeit | 20 m/s |  | Abspringpunkt |  |
| Fläche Fallschirm |  |  | Geschwindigkeit bei Landung |  |
| Gewicht Springer mit Ausrüstung |  |  |  |
| Gewünschter Landepunkt |  |  |  |  |
| Zeitpunkt Fallschirm öffnen |  |  |  |  |

Abbildung Simulation - Versuch 1

Da die Flugzeit nur 61.9 Sekunden beträgt, kommt der Springer gar nicht zum Öffnen des Fallschirmes, wodurch er mit voller Geschwindigkeit aufprallt. Dies wollen wir verhindern, wir ändern daher die Paramter. Der Fallschirm benötigt zwei Sekunden um sich vollständig zu öffnen, wir versuchen es daher einmal mit 60 Sekunden.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Simulation – Versuch 2** | | | | |
| **Paramter** | |  | **Resultate** | |
| Flughöhe |  |  | Flugzeit |  |
| Flugzeuggeschwindigkeit | 20 m/s |  | Abspringpunkt |  |
| Fläche Fallschirm |  |  | Geschwindigkeit bei Landung |  |
| Gewicht Springer mit Ausrüstung |  |  |  |
| Gewünschter Landepunkt |  |  |  |  |
| Zeitpunkt Fallschirm öffnen |  |  |  |  |

Abbildung Simulation - Versuch 2

Diese Simulation ist schon besser. Der Springer überlebt und kann eine Flugzeit von 81.2 Sekunden geniessen. Wir wollen nun aber noch mehr Risiko eingehen und öffnen den Fallschirm erst bei 61.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Simulation – Versuch 3** | | | | |
| **Paramter** | |  | **Resultate** | |
| Flughöhe |  |  | Flugzeit |  |
| Flugzeuggeschwindigkeit | 20 m/s |  | Abspringpunkt |  |
| Fläche Fallschirm |  |  | Geschwindigkeit bei Landung |  |
| Gewicht Springer mit Ausrüstung |  |  |  |
| Gewünschter Landepunkt |  |  |  |  |
| Zeitpunkt Fallschirm öffnen |  |  |  |  |

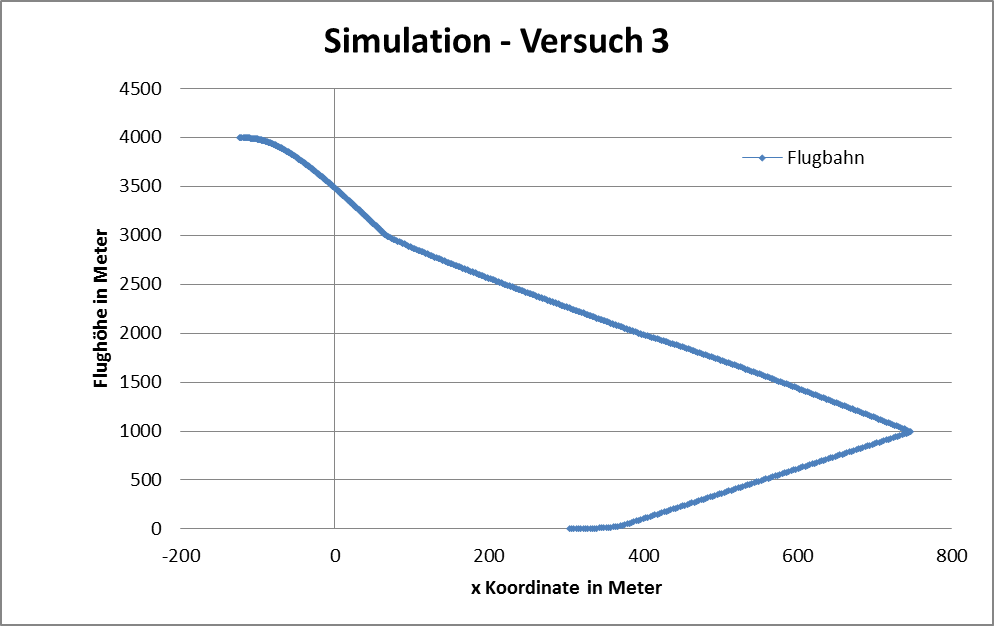


Abbildung Simulation - Versuch 3

Und es funktioniert. Theoretisch könnte der Springer den Fallschirm erst bei 61.2 Sekunden öffnen und er würde überleben, es wird dadurch eine sehr kurze Flugzeit erreicht. Wir könnten nun versuchen die Paramter zu optimieren. Beim Öffnen bei 61.3 haben wir dann aber eine Geschwindigkeit von 9.7 und bei 61.4 bereits 25m/s. In realen Bedingungen sollte man daher nicht an die Grenzen gehen und eher ein paar Sekunden früher öffnen, denn nur wenige Zehntelssekunden können bereits über Leben und Tod entscheiden.

# Fazit

## Sevi

## Patrice

## Danksagung

# Anhang

## Quellenverzeichnis

**Bärwolff, Günter. 2007.** *Numer für Ingenieure, Physiker und Informatiker.* Universität Berlin : s.n., 2007.

**Grassl, Florian.** www.fsr-club2000.de. [Online] [Zitat vom: 10. 06 2012.] http://www.fsr-club2000.de/ausbildung/Florian\_Grassl\_Facharbeit\_Freier\_Fall.pdf.

**Halliday, David. 2007.** *Halliday Physik.* 2007. ISBN: 978-3-527-40746-0.

**Kirchgraber, U. 1993.** www.educ.ethz.ch. [Online] 1993. http://www.educ.ethz.ch/unt/um/mathe/gb/Fallschirmspringer.pdf.

**Maxim. 2010.** www.virtual-maxim.de/. *Virtual Maxim.* [Online] 2010. [Zitat vom: 25. Mai 2012.] http://www.virtual-maxim.de/downloads/freier%20fall%20mit%20und%20ohne%20luftwiderstand.pdf.

**Ullenboom, Christian. 2009.** *Java ist auch eine Insel.* 2009. ISBN: 978-3-8362-1371-4.

**Wikipedia.** [Online] [Zitat vom: 10. 06 2012.] http://de.wikipedia.org/wiki/Fallschirmspringen.

## Abbildungsverzeichnis