Rakete – Satellit

Inhalt

[Einleitung 3](#_Toc323798723)

[Erde 3](#_Toc323798724)

[Satellit 3](#_Toc323798725)

[Rakete 3](#_Toc323798726)

[Abgrenzungen 3](#_Toc323798727)

[Mathematische Beziehungen 4](#_Toc323798728)

[Satellit 4](#_Toc323798729)

[Euler – Theorie 4](#_Toc323798730)

# Einleitung

Auftrag ist es, eine Simulation eines Raketenstarts und –flugs zu bauen. Die Rakete wird mit einer bestimmten Füllmenge eines Treibstoffes von einem Punkt auf der Erde starten. Das Programm soll aus den angegebenen Parametern eine Flugbahn berechnen, welche die folgenden Bedingungen erfüllt:

* Rakete kommt mit dem Treibstoff in den Orbit des Satelliten.
* Rakete ist am Ende des Fluges beim Satelliten angedockt und der Treibstoff ist verbraucht.
* Rakete und Satellit haben am Ende die gleiche Geschwindigkeit.

## Erde

Die Erde befindet sich im Zentrum des Systems. Sie hat eine definierte Masse und eine definierte Gravitation. Beide sind konstant.

## Satellit

Der Satellit befindet sich zum Startzeitpunkt in einer Umlaufbahn um die Erde. Er kreist in einer elliptischen Bewegung um die Erde. Wie die Erde ist dem Satellit eine definierte Masse zugewiesen. Ausserdem werden noch die Geschwindigkeit und die Position benötigt.

## Rakete

Die Rakete steht zu Beginn an der Abschussrampe auf der Erde. Ihr ist ebenfalls eine Masse zugewiesen. Dazu kommen noch Treibstoffmenge (variabel), Verbrennungsgrad des Treibstoffes (variabel) und Position auf der Erde. Das Programm soll nun Geschwindigkeit, Richtung und Abschusszeitpunkt berechnen. Die Rakete wird dann diesen Weg abfliegen und so zum Satelliten gelangen, an welchem sie dann andocken wird.

## Abgrenzungen

Die Körper sollen physikalisch korrekt zu einander reagieren und Effekte wie Massenanziehung beinhalten. Allerdings werden äussere Einflüsse ausgeschaltet. Sprich die Sonne und sonstige ähnliche Einflüsse werden ignoriert.

# Mathematische Beziehungen

In diesem Abschnitt möchten wir mehr auf die mathematische Seite des Projektes eingehen. Dabei wird es in Satellit und Rakete unterteilt. Beide müssen bestimmte Voraussetzungen erfüllen, damit das Programm am Schluss die korrekte Flugbahn berechnen kann.

## Satellit

Der Satellit bewegt sich in einer elliptischen Bahn um die Erde. Diese Bahn wird hauptsächlich von der Startgeschwindigkeit und der Massenanziehungskraft der beiden Körper verursacht.

Zum Zeitpunkt der Ausgangssituation kennen wir die Formel für die Berechnung der Bahn nicht. Daher muss die Ellipse angenähert werden. Diese bewerkstelligen wir mit einem Polygon. Wir werden das Polygon durch die numerische Integration nach Euler erhalten.

### Euler – Theorie

Die Integration nach Euler nähert einen Funktionsgraphen, wie schon erwähnt, durch ein Polygon an. Ein Polygon ist ein Körper, welcher aus vielen verschiedenen Datenpunkten besteht. Diese sind mit einer Geraden verbunden. Daraus folgt, je mehr Unterteilungen wir durchführen, desto genauer trifft das Annäherungspolygon den Funktionsgraphen. Allerdings brauchen wir für die Euler-Berechnung noch die Beschleunigung des Satelliten zur Zeit t.  
Der „0-Punkt“ (Punkt zur Zeit t = 0) kann durch einen Vektor in der Form (xsat, ysat, vx-sat, vy-sat). Dieser Vektor beschreibt die X- und die Y-Koordinate und die Geschwindigkeit v in X- und Y-Richtung. Für die Berechnung des neuen Punktes benötigen wir allerdings die Beschleunigung.  
Wenn wir nun den Vektor ableiten werden die Positionskoordinaten zu den Geschwindigkeiten in die jeweilige Richtung. Sprich der Vektor sieht danach so aus:

Satellit‘ = (vx-sat, vy-sat,, ax-sat, ay-sat)

Wir sehen, dass die Geschwindigkeiten des Satelliten zur Beschleunigung werden. Die Beschleunigung können wir durch die Formel

ax-sat = (-g \* m (xErde – xSat))/|u|3

ay-sat = (-g \* m (yErde – ySat))/|u|3

Nun haben wir die Informationen für Euler durch differentieren erhalten.

Euler integriert nun immer den Graphen über eine bestimmte Strecke (t0 bis tEnde) in n Teilschritten. Die Anzahl der Teilschritte bekommen wir, wenn wir unser h (Unterschied zwischen t0 und t1, t1 und t2, usw.) zuerst definieren, dann n berechnen durch (b-a)/h, dieses Ergebnis mathematisch korrekt runden (da n nur eine Ganze Zahl sein kann) und mit diesem n nochmals das korrekte h mit (b-a)/n ausrechnen.  
Nun können wir unseren neuen Punkt des Satelliten ausrechen. Den erhalten wir durch die folgende Gleichung:

Satellitneu = Satellialt + h\*Satellitalt‘

Sobald diese Rechnung durchgeführt wurde, wird t um h erhöht.

Da der Mensch diese kleinen Unterschiede bei der Darstellung gar nicht wahrnehmen könnte, wird der Satellit nur für ein klar unterscheidbares t (bis jetzt t0 + 0.1) neu gezeichnet. Somit entsteht die korrekte Flugbahn.

### Wichtige Codestellen mit Satellit

**Calculation.java**

/\*\*

\* Ableitung des aktuellen Satelliten-Vektors

\* **@param** yAnfang

\* **@param** earth

\* **@return** res

\*/

**public** **double**[] diff\_sat(**double**[] yAnfang, Earth earth)

{

**double**[] z = **new** **double**[yAnfang.length];

**double**[] res = **new** **double**[yAnfang.length];

**double**[] u = **new** **double**[2];

**double** uBetrag,g,m;

g = 10; // Gravitationskonstante

m = earth.getMass(); // Masse der Erde

**for**(**int** i=0;i<yAnfang.length;i++)

z[i] = yAnfang[i];

u[0] = z[0] - earth.getPosx();

u[1] = z[1] - earth.getPosy();

uBetrag = Math.*sqrt*(u[0]\*u[0] + u[1]\*u[1]); //Distanz zwischen Satellit und Erde

//Ableitung von x- und y-Koordinate wird zur Geschwindigkeit

res[0] = z[2];

res[1] = z[3];

//Ableitung von x- und y-Geschwindigkeit wird zur Beschleunigung

res[2] = -g \* m \* (u[0]/Math.*pow*(uBetrag, 3));

res[3] = -g \* m \* (u[1]/Math.*pow*(uBetrag, 3));

**return** res;

}

/\*\*

\* Polygon-Integration nach Euler zwischen tAnfang und tEnde mit einer

\* Aufteilung nach n. tEnde ist momentan immer tAnfang + 0.1

\* **@param** tAnfang

\* **@param** tEnde

\* **@param** yAnfang

\* **@param** n

\* **@return**

\*/

**public** **double**[] euler(**double** tAnfang,**double** tEnde,**double**[] yAnfang, **int** n)

{

**double** h = (tEnde-tAnfang)/n; //korrektes h für Unterschiede berechnen

**double**[] y = yAnfang;

**double**[] k;

**double** t = tAnfang;

**for**(**int** i=1;i<=n;i++)

{

k = diff\_sat(y,**new** Earth()); //Ableitung des momentanen Vektors

y = addVector(y,multScalarVector(h,k)); //neuer Vektor berechnen

t = t+h;

}

**return** y;

}