TEKNOFEST 2022 ROKET YARIŞMASI

Orta İrtifa Kategorisi

Uçuş Benzetim Raporu (UBR)

MBK KUZGUN ROKET TAKIMI

İçindekiler

1 Sorular ve Cevaplar

1.1 Cevap 1

Kinematik, dinamiğin bir alt dalıdır. Kinematik, hareketin geometrisini inceleyen bir dal olup hareketin sebebinden bağımsız bir şekilde yerdeğiştirme, hız, ivme ve zaman arasında ilişkiler kurarak hareketini mukayese eden dalıdır. Dinamik iki alt dala ayırılmaktadır: Kinematik ve Kinetik. Kinetik ise cisme etki eden kuvvet(ler), cismin kütlesi ve cismin hareketi arasında ilişkiler kurarak cismin hareketin nasıl olacağını yada cismin hareketi istenildiği gibi yönetmek için uygulanması gereken kuvvet(ler)i inceleyen daldır.[1]

1.2 Cevap 2

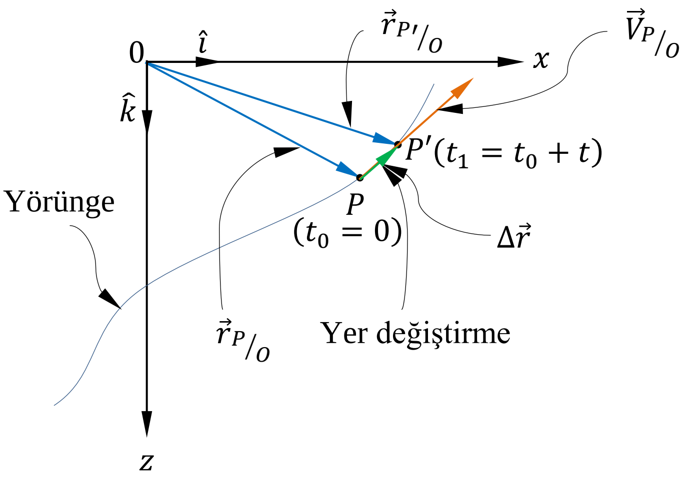
Roket için motor seçimi, roketin gövdenin ölçüleri, burun konisinin seçilmesi ve ölçülendirimesi, kanatçık seçimi ve ölçülendirilmesi gibi işlemler yapmamızı sağlamaktadır. Kısacası bir roketin ilk genel şeklini belirlememizi sağlamakta olup tasarımın detayına girildiği zaman mukavemet, akışkanların dinamiği gibi alanlar yönünden çalışma yaparken sınır şartlar ve başlangıç koşullarını belirlenmesini sağlamaktadır.

1.3 Cevap 3

Roketin dönmesi, irtifa ve menzili diğer bir ifadeyle roketin hareket edeceği yönünü etki etmektedir. Açısal hareketiyle birlikte modellenirse roketin gideceğe yönü gerçeğe daha yakın olup bizim istediğimiz gibi uçacağa şeklinde tasarlanılabilcektir. Bunu gerçekleştirmek için roketin kütle atalet momenti bilinmesi gerekir ve bir atmosferik model oluşturulması gerekmektedir. Bunun yanında rüzgar da modellenirse daha doğru sonuç verecektir.

2 Kinematik Denklemler

2.1 Yer değiştirme, hız ve ivme arasındaki ilişki



Şekil 2.1.1

Şekil 2.1.1, bir partikülün (maddesel noktanın) yaptığı 2 boyutlu hareketin yörünge üzerinde yer değiştirme ve hız arasındaki ilişkisini göstermektedir. Şekilden 2.1.1'den yer değiştirme, hız ve ivme arasındaki matematiksel ilişkiler kurulabilmektedir.

Bir partikülün yer değiştirme vektörü;

$$\vec{r}_{P/0} = \vec{r}_x(t) + \vec{r}_z(t)$$

$$\vec{r}_{P/0} = \vec{x}(t) + \vec{z}(t)$$

$$\vec{r}_{P/0} = x(t)\hat{\imath} + z(t)\hat{k} \quad \text{.......} \quad (2.1.1)$$

Bir partikülün hız vektörü;

$$\vec{V}_P(t) = \lim_{\Delta t \to 0} \frac{\Delta \vec{r}_P}{\Delta t}$$

$$\vec{V}_P(t) = \frac{d(\vec{r}_P)}{dt}$$

$$\vec{V}_P(t) = \frac{d\big(x(t)\hat{\imath} + z(t)\hat{k}\big)}{dt}$$

$$\vec{V}_P(t) = \dot{x}(t)\hat{\imath} + x(t)\frac{d\hat{\imath}}{dt} + \dot{z}(t)\hat{k} + z(t)\frac{d\hat{k}}{dt}$$

Not:

$$\dot{x} = \frac{dx}{dt}$$

$$\dot{z} = \frac{dz}{dt}$$

Dönme hareket olmayacağından;

$\frac{d\hat{\imath}}{dt} = 0$ ve $\frac{d\hat{k}}{dt} = 0$ olacaktır. O halde;

$$\vec{V}_P(t) = \dot{x}(t)\hat{\imath} + \dot{z}(t)\hat{k}$$

$$\vec{V}_P(t) = v_x(t)\hat{\imath} + v_z(t)\hat{k} \quad \text{.......} \quad (2.1.2)$$

Bir partikülün ivme vektörü;

Bir partikülün hız vektörü yörüngeye teğettir fakat ivme vektörün yörüngeye teğet olup olmadığını ilgili kesin bir şey söylenemez. Çünkü ivme dediğimiz zaman;

a: bileşke ivmesi

a_t: teğetsel ivmesi

a_n: normal ivmesi

$$a = \sqrt{a_t^2 + a_n^2}$$

Böylekikle 3 farklı ivme olduğu unutulmamalıdır. Fakat şekil 2.1.1'de gösterilen hareketin bileşkeleri 2 ayrı doğrusal hareket olarak kabul edilirse yalnız teğetsel ivme olup teğetsel ivme daima yörüngeye teğet olacaktır. Bir partikülün doğrusal hareket esnasında ivme vektörü;

$$\vec{a}_P(t) = \lim_{\Delta t \to 0} \frac{\Delta \vec{V}_P}{\Delta t}$$

$$\vec{a}_P(t) = \frac{d\vec{V}_P}{dt}$$

$$\vec{a}_P(t) = \frac{d\left(v_x(t)\hat{\imath} + v_z(t)\hat{k}\right)}{dt}$$

$$\vec{a}_P(t) = \dot{v}_x(t)\hat{\imath} + v_x(t)\frac{d\hat{\imath}}{dt} + \dot{v}_z(t)\hat{k} + v_Z(t)\frac{d\hat{k}}{dt}$$

Not:

$$\dot{v}_x = \frac{dv_x}{dt}$$

$$\dot{v}_z = \frac{dv_Z}{dt}$$

Doğrusal hareket kabul edildiğinden;

$\frac{d\hat{\imath}}{dt} = 0$ ve $\frac{d\hat{k}}{dt} = 0$ olacaktır. O halde;

$$\vec{a}_P(t) = \dot{v}_x(t)\hat{\imath} + \dot{v}_z(t)\hat{k}$$

$$\vec{a}_P(t) = a_x(t)\hat{\imath} + a_z(t)\hat{k} \quad \text{.......} (2.1.3)$$

İvme vektörün bileşkeleri ayrı ayrı yazılırsa;

$$a_x(t) = \frac{dv_x}{dt}$$

$$dv_x = a_x(t)dt$$

$$\int_{v_{x_0}}^{v_x(t)} dv_x = \int_0^t a_x(t)dt$$

$$[v_x]_{v_{x0}}^{v_x(t)} = \int_0^t a_x(t)dt$$

$$v_x(t) - v_{x_0} = \int_0^t a_x(t)dt$$

$$v_x(t) = v_{x_0} + \int_0^t a_x(t)dt \quad \text{.......} (2.1.4)$$

$$v_x(t) = \frac{dx}{dt}$$

$$a_z(t) = \frac{dv_z}{dt}$$

$$dv_z = a_z(t)dt$$

$$\int_{v_{z_0}}^{v_z(t)} dv_z = \int_0^t a_z(t)dt$$

$$[v_z]_{v_{z0}}^{v_z(t)} = \int_0^t a_z(t)dt$$

$$v_z(t) - v_{z_0} = \int_0^t a_z(t)dt$$

$$v_z(t) = v_{z_0} + \int_0^t a_z(t)dt \quad \text{.......} (2.1.6)$$

$$v_z(t) = \frac{dz}{dt}$$

$dx = v_x(t)dt$

$\int_{x_0}^{x(t)} dx = \int_0^t v_x(t)dt$

$[dx]_{x_0}^{x(t)} = \int_0^t v_x(t)dt$

$x(t) - x_0 = \int_0^t v_x(t)dt$

$x(t) = x_0 + \int_0^t v_x(t)dt$ ....(2.1.5)

$dz = v_z(t)dt$

$\int_{z_0}^{z(t)} dz = \int_0^t v_z(t)dt$

$[dz]_{z_0}^{z(t)} = \int_0^t v_z(t)dt$

$z(t) - z_0 = \int_0^t v_z(t)dt$

$z(t) = z_0 + \int_0^t v_z(t)dt$.................... (2.1.7)

2.2 Uçuş yolu açısı

Uçuş yolu açısını elde etmek için yatay yani x ekseni ile yaptığı açısı kabul edilirse, partikülün z doğrultusunda x'e göre değişmi bulunursa trigonometrik bağlantılarıyla açıya ulaşılabilir. Bu cümle matematiksel olarak gösterilirse;

$\tan(\theta) = \frac{dz(t)}{dx(t)}$

$\theta = \arctan\left(\frac{dz(t)}{dx(t)}\right)$ .... (2.2.1)

Denklem 2.2.1'de olan $\frac{dz(t)}{dx(t)}$ ifadesi doğrudan bulunamamaktadır. Bunun sebebi ise $z(t)$ ve $x(t)$ fonksiyonları nümerik integrasonlar yapılarak elde edildiğinden ayrık fonksiyonlardır (discrete functions). Fakat $\frac{dz(t)}{dx(t)}$ ifadesine zincir kuralından yararlanarak aşağıdaki denklem 2.2.2 gibi yazılırsa;

$\theta = \arctan\left(\frac{dz(t)}{dt}\,\frac{dt}{dx(t)}\right)$ .... (2.2.2)

$\theta = \arctan\left(\frac{\frac{dz(t)}{dt}}{\frac{dx(t)}{dt}}\right)$

$\theta(t) = \arctan\left(\frac{v_z(t)}{v_x(t)}\right)$ .... (2.2.3)

Denklem 2.2.3 karşımıza çıkmaktadır ve bu denkleme göre uçuş yolu açısı doğrudan partikülün hız bileşkelerinden elde edilebildiğini göstermektedir. Çözüm esnasında $v_z(t)$ ve $v_x(t)$ ifadelerin sayısal değerleri bulunduğundan bu şekilde hesaplamak daha uygun ve daha doğru sonuç verecektir.

3 Uçuş Benzetimi

Denklem 2.1.4, 2.1.5, 2.1.6 ve 2.1.7 integral denklemler olup, bu denklemler nümerik integrasyon yöntemleriyle çözülmesi gerekmektedir. Denklem 2.1.4 ve 2.1.6 Simpson's $1/3^{rd}$ Rule yöntemiyle ve denklem 2.1.5 ve 2.1.7 ise 2. (ikinci) dereceden polinomla interpolasyon uygulanarak ayrık fonksiyon integrasyon yöntemle çözülecektir.

3.1 Simpson's 1/3rd Rule of Integration [2]

Simpson's $1/3^{rd}$ Rule'ün denklemi katsayılar yöntemiyle çıkartılırsa;

$$\int_a^b f(x)dx \approx c_1 f(a) + c_2 f\left(\frac{a+b}{2}\right) + c_3 f(b) \quad (3.1.1)$$

Eğer denklemin sağ tarafı $\int_a^b 1dx$, $\int_a^b xdx$ and $\int_a^b x^2dx$ bu integrallara eşit kabul edilirse;

$$\int_a^b 1dx = b - a = c_1 + c_2 + c_3 \quad (3.1.2)$$

$$\int_a^b xdx = \frac{b^2-a^2}{2} = c_1 a + c_2 \frac{a+b}{2} + c_3 b \quad (3.1.3)$$

$$\int_a^b x^2dx = \frac{b^3-a^3}{3} = c_1 a^2 + c_2 \left(\frac{a+b}{2}\right)^2 + c_3 b^2 \quad (3.1.4)$$

Denklem 3.1.2, 3.1.3 ve 3.1.4 c_0, c_1 ve c_2 katsayıları elde etmek için çözülürse;

$$c_1 = \frac{b-a}{6} \qquad c_2 = \frac{2(b-a)}{3} \qquad c_3 = \frac{b-a}{6}$$

O halde denklem 3.1.1;

$$\int_a^b f(x)dx \approx \frac{b-a}{6} f(a) + \frac{2(b-a)}{3} f\left(\frac{a+b}{2}\right) + \frac{b-a}{6} f(b)$$

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \quad (3.1.4)$$

3.2 2. Dereceden polinom ile interpolasyon yaparak ayrık fonksiyonların integrasyonu [3]

2. dereceden polinomla interpolasyon yapılabilmesi için bir 2. dereceden polinom denkleminde olan 3 adet katsayılarını bulmak için 3 adet denkleme ihtiyaç duyulmaktadır. Matematiksel ifadelerle gösterilirse;

x	$f(x)$
x_0	$f(x_0)$
x_1	$f(x_1)$
x_2	$f(x_2)$

Tablo 3.2.1

$I = \int_a^b f(x)dx$, $\{x_0 \leq a, x_2 \geq b\}$ .... (3.2.1)

$f(x) = a_0 + a_1x + a_2x^2$

Bilinmeyen katsayılarını bulmak için Tablo 3.2.1'dan 3 denklem oluşturulur;

$$\left.\begin{aligned} f(x_0) &= a_0 + a_1x_0 + a_2x_0^2 \\ f(x_1) &= a_0 + a_1x_1 + a_2x_1^2 \\ f(x_2) &= a_0 + a_1x_2 + a_2x_2^2 \end{aligned}\right\} \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \end{bmatrix}$$

Denklem 3.2.2

Oluşturulan 3 adet denklemi, denklem 3.2.2 formatında yazarak Gauss Eleme Yöntemi gibi lineer denklem seti çözebilen bir yöntem uygulanarak bilinmeyen a_0, a_1 ve a_2 katsayılar bulunur. Bu katsayıların sayısal değerler eldedildiğinde denklem 3.2.1'e analitik olarak integrasyon yapılırsa;

$$I = \int_a^b a_0 + a_1x + a_2x^2 dx$$

$$I = \left[a_0x + \frac{a_1x^2}{2} + \frac{a_2x^3}{3}\right]_a^b$$

$I = a_0(b-a) + \frac{a_1(b^2-a^2)}{2} + \frac{a_2(b^3-a^3)}{3}$ .... (3.2.3)

Denklem 3.2.3 ayrık fonksiyonları 2. dereceden polinomla interpolasyon yaparak integrasyon yapmamızı sağlamaktadır.

3.3 Nümerik kinematik denklemler

Denklem 2.1.4, 2.1.5, 2.1.6 ve 2.1.7 yi nümerik bir şekilde çözmek için bölüm 3.2'de anlatılmış olan nümerik integrasyon yöntemler uygulandığında;

Denklem 2.1.4 ve 3.1.4'den;

$$v_x(t) \approx v_{x_0} + \sum_{i=0}^{i=n} \frac{b-a}{6}\left(a_x(a_i) + 4a_x\left(\frac{a_i+b_i}{2}\right) + a_x(b_i)\right) \quad (3.3.1)$$

Denklem 3.3.1'de $b - a = \Delta t$ (Δt: zaman adımı), $a = t_0$ ve $b = t_1$olarak yazılırsa;

$$v_x(t) \approx v_{x_0} + \sum_{i=0}^{i=n} \frac{\Delta t}{6}\left(a_x(t_{0_i}) + 4a_x\left(\frac{t_{0_i}+t_{1_i}}{2}\right) + a_x(t_{1_i})\right) \quad (3.3.2)$$

Denklem 3.3.2 partikülün x yönündeki hız bileşenini veren denklemdir. $n = \Delta t \times t$ dir. Aynı şekilde z yönündeki hız bileşeni için denklem 2.1.6 ve 3.1.4'den;

$$v_z(t) \approx v_{z_0} + \sum_{i=0}^{i=n} \frac{b-a}{6}\left(a_z(a_i) + 4a_z\left(\frac{a_i+b_i}{2}\right) + a_z(b_i)\right) \quad (3.3.3)$$

Denklem 3.3.3'de $b - a = \Delta t$ (Δt: zaman adımı), $a = t_0$ ve $b = t_1$olarak yazılırsa;

$$v_z(t) \approx v_{z_0} + \sum_{i=0}^{i=n} \frac{\Delta t}{6}\left(a_z(t_{0_i}) + 4a_z\left(\frac{t_{0_i}+t_{1_i}}{2}\right) + a_z(t_{1_i})\right) \quad (3.3.4)$$

Denklem 3.3.4 z yönündeki hız bileşeni veren denklemdir. Denklem 3.3.2 ve 3.3.4 farklı noktalrdaki sayısal değerini verdiği için, diğer bir ifadeyle ayrık fonksiyon olduğundan yer değiştirme bileşenini elde etmek için denklem 2.1.5 ve 3.2.3'ten;

$$x(t) \approx x_0 + \sum_{i=0}^{i=n} \left(a_{0x_i}(b-a) + \frac{a_{1x_i}(b_i^2 - a_i^2)}{2} + \frac{a_{2x_i}(b_i^3 - a_i^3)}{3}\right) \quad (3.3.5)$$

Denklem 3.3.5'de $a = t_0$ ve $b = t_1$ ($t_1 - t_0 = \Delta t$ (Δt: zaman adımı)) olarak yazılırsa;

$$x(t) \approx x_0 + \sum_{i=0}^{i=n} \left(a_{0x_i}(t_{1_i} - t_{0_i}) + \frac{a_{1x_i}(t_{1_i}^2 - t_{0_i}^2)}{2} + \frac{a_{2x_i}(t_{1_i}^3 - t_{0_i}^3)}{3}\right) \quad (3.3.6)$$

Denklem 3.3.6 partikülün x bileşenin yerdeğiştirmesini vermektedir. Aynı şekilde z bileşenin denklemini elde etmek için denklem 2.1.7 ve 3.2.3 den;

$$z(t) \approx z_0 + \sum_{i=0}^{i=n} \left(a_{0z_i}(b_i - a_i) + \frac{a_{1z_i}(b_i^2 - a_i^2)}{2} + \frac{a_{2z_i}(b_i^3 - a_i^3)}{3}\right) \quad (3.3.7)$$

Denklem 3.3.5'de $a = t_0$ ve $b = t_1$ ($t_1 - t_0 = \Delta t$ (Δt: zaman adımı)) olarak yazılırsa;

$$z(t) \approx z_0 + \sum_{i=0}^{i=n} \left(a_{0z_i}\left(t_{1_i} - t_{0_i}\right) + \frac{a_{1z_i}\left(t_{1_i}^2 - t_{0_i}^2\right)}{2} + \frac{a_{2z_i}\left(t_{1_i}^3 - t_{0_i}^3\right)}{3} \right) \quad (3.3.8)$$

Denklem 3.3.5 ve 3.3.6 daki katsayılarını (a_{0x}, a_{1x} ve a_{2x}) ve 3.3.7 ve 3.3.8 deki katsayılarını (a_{0z}, a_{1z} ve a_{2z}) bulmak için oluşturluması gereken denklemlerin verileri tablo halde aşağıda göstermektedir.

t	$v_x(t)$	$v_z(t)$
t_0	$v_x(t_0)$	$v_z(t_0)$
t_1	$v_x(t_1)$	$v_z(t_1)$
t_2	$v_x(t_2)$	$v_z(t_2)$

Tablo 3.3.1

Bilinmeyen katsayılarını çözmek için lineer denklem set olarak matriste gösteririse;

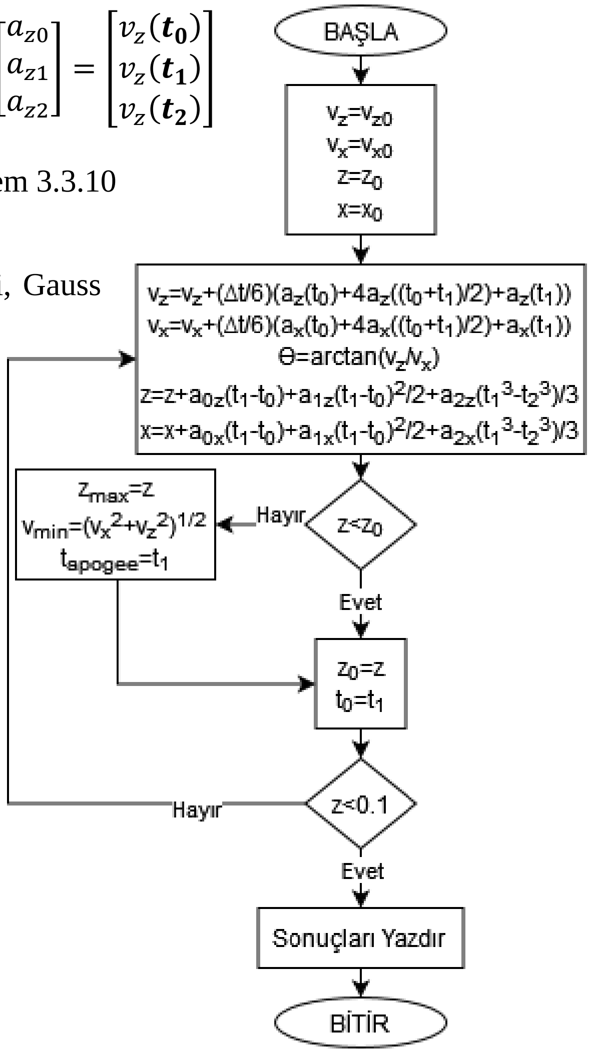
$$\begin{bmatrix} 1 & t_0 & t_0^2 \\ 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \end{bmatrix} \begin{bmatrix} a_{x0} \\ a_{x1} \\ a_{x2} \end{bmatrix} = \begin{bmatrix} v_x(t_0) \\ v_x(t_1) \\ v_x(t_2) \end{bmatrix}$$

Denklem 3.3.9

$$\begin{bmatrix} 1 & t_0 & t_0^2 \\ 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \end{bmatrix} \begin{bmatrix} a_{z0} \\ a_{z1} \\ a_{z2} \end{bmatrix} = \begin{bmatrix} v_z(t_0) \\ v_z(t_1) \\ v_z(t_2) \end{bmatrix}$$

Denklem 3.3.10

Denklem 3.3.9 ve 3.3.10 Gauss eleme yöntemi, Gauss Seidel gibi lineer denklem sistem çözebilen yöntemleriyle çözülerek bilinmeyen katsaylar bulunur. Bu raporda bu katsayılarını bulmak için Gauss eleme yöntemi kullanımıştır. Denklemleri çözmek için Java yazılım dilinde bir kod oluşturarak çözülmüştür. Kodun akış diyagramı şekil 3.3.1'de verilmiştir. Java kodu da aşağıda verilmiştir.



Şekil 3.3.1

```
package sample;

import javafx.fxml.Initializable;
import java.net.URL;
import java.util.ResourceBundle;

public class Controller implements Initializable {
    double   z_0    =           0;
    double   x_0    =           0;
    double   vz_0   =     -93.969;
    double   vx_0   =      34.202;
    double   theta  =          70;
    double[] z      =   {0,0,z_0};
    double[] x      =   {0,0,x_0};
    double[] v_z    =  {0,0,vz_0};
    double[] v_x    =  {0,0,vx_0};
    double[] t      =     {0,0,0};
    double[] a      =     {0,0,0};
    double delta_t  =        0.01;
    double   max_alt=z_0,min_vel,max_alt_time;
    @Override
    public void initialize(URL location, ResourceBundle resources) {
        do{
            simulate();
            System.out.println(String.format("t: %.3f\tv_z: %.3f\tz: %.3f\tv_x: %.3f\tx: %.3f\tθ: %.3f",t[2],v_z[2],z[2],v_x[2],x[2],theta));
        }while(z[2]<0.1);
        printSummary();
    }
    double a_z (double t){
        return 9.801;
    }
    double a_x (double t){
        return 0;
    }
    double I_z (double a, double b){
        return ((b-a)/6)*(a_z(a)+(4*a_z((a+b)/2))+a_z(b));
    }
    double I_x(double a, double b){
        return ((b-a)/6)*(a_x(a)+(4*a_x((a+b)/2))+a_x(b));
    }

    void simulate(){
        calc_velocity(delta_t);
        calc_flight_angle(v_z[2],v_x[2]);
        calc_displacement(t[0],v_z[0],t[1],v_z[1],t[2],v_z[2],'z');
        calc_displacement(t[0],v_x[0],t[1],v_x[1],t[2],v_x[2],'x');
        if(z[2]<max_alt){
            max_alt=z[2];
            min_vel = Math.sqrt(Math.pow(v_z[2],2)+Math.pow(v_x[2],2));
            max_alt_time = t[2];
        }

    }
    void calc_velocity (double delta_t){
        t[0]=t[1];
        t[1]=t[2];

```

```
        v_z[0]=v_z[1];
        v_z[1]=v_z[2];

        v_x[0]=v_x[1];
        v_x[1]=v_x[2];

        t[2]=t[1]+delta_t;

        v_z[2]  =   v_z [1] + I_z(t[1],t[2]);
        v_x[2]  =   v_x [1] + I_x(t[1],t[2]);
    }
    void calc_displacement (double t_0, double v_0, double t_1, double v_1, double t_2, double v_2, char component){
        double mat[][] = {  { 1, t_0, Math.pow(t_0,2),v_0}
,
                            { 1, t_1, Math.pow(t_1,2),v_1
},
                            { 1, t[2], Math.pow(t_2,2),v_2
}
                          };
        gaussianElimination(mat);
        switch (component){
            case 'z' :  z[0] = z[1];
                        z[1] = z[2];
                        z[2] = z[1] + (a[0]*(t[2]-t[1])) + (a[1]*((Math.pow(t[2],2)-Math.pow(t[1],2))/2)) + (a[2]*((Math.pow(t[2],3)-Math.pow(t[1],3))/3));
                        break;
            case 'x' :  x[0] = x[1];
                        x[1] = x[2];
                        x[2] = x[1] + (a[0]*(t[2]-t[1])) + (a[1]*((Math.pow(t[2],2)-Math.pow(t[1],2))/2)) + (a[2]*((Math.pow(t[2],3)-Math.pow(t[1],3))/3));
                        break;
        }

    }
    void calc_flight_angle(double v_z, double v_x){
        theta = Math.toDegrees(Math.atan2(v_z,v_x));
    }
    void printSummary(){
        max_alt*=-1;
        System.out.println("\n");
        System.out.println(String.format("Maximum Altitude              : %.3f [m]",max_alt));
        System.out.println(String.format("Velocity at maximum altitude : %.3f [m/s]",min_vel));
        System.out.println(String.format("Time at maximum altitude     : %.3f [s]",max_alt_time));
        System.out.println(String.format("Final Position               : %.3f,0,%.3f [m]",x[2],z[2]));
        System.out.println(String.format("Final velocity               : %.3f [m/s]",Math.sqrt(Math.pow(v_z[2],2)+Math.pow(v_x[2],2))));
        System.out.println(String.format("Final flight angle           : %.3f [°]",theta,t[2]));
        System.out.println(String.format("Final time                   : %.3f [s]",t[2]));
    }
    public static int N = 3;
    void gaussianElimination(double mat[][]) {
        int singular_flag = forwardElim(mat);
        if (singular_flag != -1) {
```

```
105                 if (mat[singular_flag][N] != 0) {
106                 }
107                 else {
108                 }
109 
110                 return;
111             }
112             backSub(mat);
113         }
114         void swap_row(double mat[][], int i, int j) {
115             for (int k = 0; k <= N; k++) {
116                 double temp = mat[i][k];
117                 mat[i][k] = mat[j][k];
118                 mat[j][k] = temp;
119             }
120         }
121         int forwardElim(double mat[][]){
122             for (int k = 0; k < N; k++){
123                 int i_max = k;
124                 int v_max = (int)mat[i_max][k];
125                 for (int i = k + 1; i < N; i++)
126                     if (Math.abs(mat[i][k]) > v_max) {
127                         v_max = (int)mat[i][k];
128                         i_max = i;
129                     }
130                 if (mat[k][i_max] == 0)
131                     return k;
132                 if (i_max != k){
133                     swap_row(mat, k, i_max);
134                 }
135                 for (int i = k + 1; i < N; i++) {
136                     double f = mat[i][k] / mat[k][k];
137                     for (int j = k + 1; j <= N; j++)
138                         mat[i][j] -= mat[k][j] * f;
139                     mat[i][k] = 0;
140                 }
141             }
142             return -1;
143         }
144         void backSub(double mat[][]) {
145             double x[]
146                     = new double[N];
147             for (int i = N - 1; i >= 0; i--) {
148                 x[i] = mat[i][N];
149                 for (int j = i + 1; j < N; j++) {
150                     x[i] -= mat[i][j] * x[j];
151                 }
152                 x[i] = x[i] / mat[i][i];
153             }
154             for (int i = 0; i < N; i++)
155             {
156                 a[i] = x[i];
157             }
158         }
159 }
```

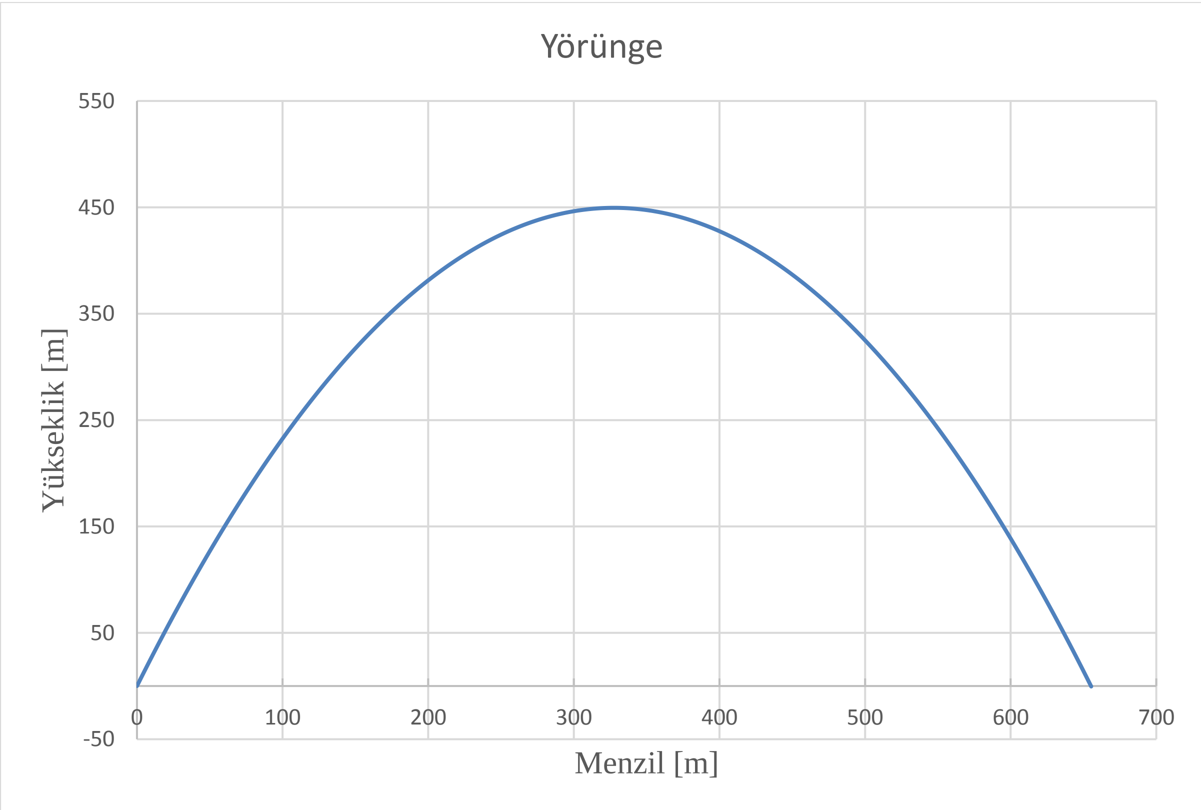
Bu kodda satır 101-153 arasındaki kod Gauss eleme yöntemi algoritması [4] olup bu kodun referansı referans kısmında belirtilmiştir. Bu kod bir JavaFX uygulaması olarak geliştirilmiştir.

Geliştirmek için IntelliJ IDEA 2018.2.4 programını kullanarak JDK 1.8.0 kullanılmıştır. Elde edilen verinin grafiğini çizdirmek için Microsoft Excel kullanılmıştır. Bunun sebebi ise Java kodun kısa tutmak içindir.

4 Benzetimin Doğrulanması

Benzetım Çıktısı	
	Değer
Tepe Noktası Yüksekliği [m]	449.534
Tepe Noktası Hızı (bileşke) [ms^{-1}]	34.202
Tepe Noktası Zamanı [s]	9.590
Son Pozisyon [m]	[655.310 , 0 , 0.433]
Son Hız (bileşke) [ms^{-1}]	99.950
Son Uçuş Yolu Açısı [°]	69.990
Son Uçuş Zamanı [s]	19.170

Tablo 4.1



Şekil 4.1

KAYNAKLAR

1 – Beer-Johnston-Cornwell. “Vector Mechanics for Engineers”, McGraw-Hill Primis, 9th edition

2 – Steven C. Chapra & Raymond P. Canale. “Numerical Methods for Engineers”, 2015

3 – Autar Kaw. “Integrating discrete functions”,2009

4 – Dharanendra L V. https://www.geeksforgeeks.org/gaussian-elimination/?ref=gcse