TEKNOFEST
HAVACILIK, UZAY VE TEKNOLOJİ FESTİVALİ

HANGAR ROKET TAKIMI
KRİTİK TASARIM RAPORU (KTR)
UÇUŞ BENZETİMİ

MAYIS 2022

TERMİNOLOJİ

A	area	**ms**	stage structural mass
a	acceleration	**mt**	total mass of rocket
α	angle of attack	**n**	number of steps
c	speed of sound	**nos**	total number of stages
CA	axial force coefficient	**ADD**	Adi Diferansiyel Denklem
CA,b	base axial force coefficient	**P**	pressure
CA,f	forward axial force coefficient	**q**	dynamic pressure
CD	drag coefficient	**Re**	Reynolds number
CN	normal force coefficient	**RE**	radius of Earth
D	drag force	**Rg**	gas constant
Df	diameter of fuselage	**Rr**	radius of launch rod
dt	time differential	**ρ**	density
δ	angle of deflection	**ρ0**	sea-level density
E	Young's modulus of elasticity	**T**	temperature
F	thrust	**TF**	thrust force
g	gravitational acceleration	**Tstage**	stage thrust
g0	sea-level gravitational acceleration	**t**	time
γ	specific hear ratio	**tb**	burn time
I	moment of inertia	**ts**	time span twait
kd	ballistic coefficient	**ty**	time to reach apogee in x-y frame
kt	'thrust coefficient'	**tz**	time to reach apogee in par-per frame
L	lapse rate	**θ**	flight angle θd
Lr	length of launch rod	**θi**	initial flight angle
Ls	length of rocket structure	**u**	horizontal component of velocity
M	Mach number	**v**	vertical component of velocity
Mend	final Mach value in drag profile	**V**	velocity Vcr
m	mass during flight	**Vx**	horizontal component of velocity
md	mass flow rate	**Vy**	vertical component of velocity
mf	final stage mass	**W**	weight x
mi	initial stage mass	**Xx**	landing distance y
tx	time to reach distance	**Yx**	apogee in x-y frame z
mp	stage propellant mass	**Zx**	

İÇİNDEKİLER

1. KİNEMATİK ve DİNAMİK DENKLEMLER

Kinematik ve Dinamik denklemlerde hız, uçuş yolu, konum ve ivme formülleri Runge-Kutta ve Euler denklemleri ile hazırlanmıştır.

2.1 Hız Denklemleri

Roketlerin sabit uçuş yapabilmesi için belirli bir hıza ulaşması gerekir. Açıyı "kritik" bir hız olan Vcr' ye ulaşılana kadar sabit tutmak makul bir çözüm olabilmektedir.

Kritik hızda tahmin elde etmek için ilk harekete Newton'un 2. yasası uygulanmıştır. Burada sürükleme ihmal edilebilir düzeydedir ve fırlatma açısı düşükse yerçekimi neredeyse hız vektörüne diktir.

$m * a$ = kuvvet (eq 1.1)

Yeniden düzenleyerek elde ederiz.

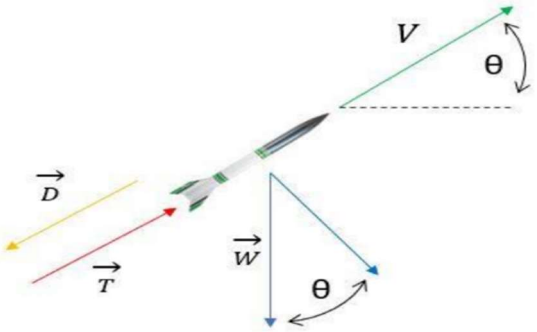
$$\frac{dv}{dt} = \frac{T_F}{m} \quad \text{(eq 1.2)}$$

Buradan, bize kullanılacak bir denklem sağlamak için aşağıdaki entegrasyonu gerçekleştiririz.

$$\int_0^v VdV = \int_0^L \frac{T_\Gamma}{m} dx \quad \text{(eq 1.3)}$$

Hızımızın sıfırdan mevcut hıza entegre edildiği ve itme-kütle oranının bir mesafenin uzunluğu boyunca entegre edildiği durumda durumda sıfırdan L'ye, burada L değeri roket gövdesi uzunluğu kabul edilmektedir. Buna ek olarak çok kısa süre boyunca kütle sabit kabul edilmektedir. Gerçekleştirme entegrasyonu ile denklemin son hali bulunur.

$$\mathrm{Vcr} = \sqrt{2 \cdot \frac{\mathrm{TF}}{\mathrm{m}} \cdot \mathrm{L}} \quad \text{(eq 1.4)}$$



Şekil 1.1 Paralel-dik koordinat çerçevesinin kuvvetlerin gösterimi

1. dereceden Runge-Kutta yöntemi olarak da bilinen Euler yöntemi, daha basit olanıdır. Denklemleri basit ve takip etmesi kolay tutmak için bazı formül ayrıştırılmaları yapılmıştır.

$$k_d = \frac{p \cdot C_D \cdot A}{2m} \qquad \text{(eq 1.5)}$$

Bu değişken bir balistik katsayı biçimidir. Bu formül ile, benzer bir amacı doldurmak için bir itme katsayısı oluşturuldu.

$$k_t = \frac{TF}{m} \qquad \text{(eq 1.6)}$$

Bu denklemler hız denklemine uyarlanmıştır.

$$\frac{dV}{dt} = \frac{TF}{m} - \frac{D}{m} - g \cdot \cos\theta \qquad \text{(eq 1.7)}$$

Euler Yöntemine göre

$$\frac{dV}{dt} = F(t, v) \Rightarrow V(t_0) = V_0 \qquad \text{(eq 1.8)}$$

Başlangıç değer problemi olarak hız formülünü ele alırsak

$$\frac{dV}{dt}$$

Türev işlemi için geri fark formülü kullanıla bilir.

$$\frac{dV}{dt} = \frac{V_i - V_{i-1}}{t_i - t_{i-1}} \qquad \text{(eq 1.9)}$$

Buradaki $t_i - t_{i-1}$ zaman farkı yerine dt'yi işlem adım değişkeni olarak yerine koyarsak;

$$\frac{dV}{dt} = \frac{V_i - V_{i-1}}{dt} \qquad \text{(eq 1.10)}$$

olur

$$\frac{V_i - V_{i-1}}{dt} = \frac{TF}{m} - \frac{D}{m} - g \cdot \cos\theta \quad \text{(eq 1.11)}$$

$$v_i = v_{i-1} + \left[\frac{TF}{m} - \frac{D}{m} - g \cdot \cos\theta\right] dt \quad \text{(eq 1.12)}$$

şeklinde numerik ifade elde ederiz .

Bu denklemde Drog(D) katsayısı

$$\mathrm{D} = \frac{1}{2} p . V^2 . c_d . A \quad \text{(eq 1.13)}$$

olarak alınırsa ve

$$k_t = \frac{TF}{m}\,,\; k_d = \frac{p \cdot C_D \cdot A}{2m} \quad \text{(eq 1.14)}$$

değişken değiştirme ile yapılırsa

$$V_i = V_{i-1} + \left[k_t - k_d (V_{i-1})^2 - g . \cos\left(\theta_{(i-1)}\right)\right] dt \quad \text{(eq 1.15)}$$

olarak numerik hız denklemi bulunur.

2.2 Uçuş Yolu Açısı Denklemi

Mevcut zaman adımındaki değer, bunu belirlemek için eşitlik 1.15'de görüldüğü gibi hız değerini kullanır. Bu paralel-dikey çözüm komut dosyasının trigonometrisi kullandığını da not etmek önemlidir. Açının yatay yerine göre dikeyden hesaplanması gibi işlevler bu formül aracılığı ile yapılır.

Uçuş Yolu Açısı Hesabı 1. Mertebe Euler Denklemlerine göre hız denklemindeki formüllere benzer işlemler sonucu eşitlik 1.16'da görüldüğü gibi bulunmuştur.

$$\theta(i) = \left(\frac{g}{V_{\{i\}}} \cdot \sin\big(\theta(i-1)\big)\right) \cdot dt + \theta(i-1) \quad \text{(eq 1.16)}$$

Denklemin yerçekimi ivmesini yeni hıza böldüğü yerde, zaman adımı ve önceki değere eklenmektedir. Uçuş açısının düşmeyeceği şekilde uçuşun ilk aşamalarında açı düşüşünün düzeltilmesi belirli koşullar oluşursa değişir. Bu koşullar şunlardır: hız değeri eşittir veya daha azdır.

Fırlatma kuvveti veya hız değeri Vcr değişkeninin değerini aşmadıysa bir motorun yanma süresi sırasında sıfırdan büyükse, kat edilen mesafe, motorun uzunluğundan daha az olmaktadır.

2.3 Konum Denklemleri

Konum denklemleri 1st Mertebe Euler Denklemlerine göre uçuş hızı, uçuş mesafesi ve uçuş yolu açısının birbirine eklenmesi sonucu oluşmuştur.

$$x(i) = \big(V(i) * \sin\big(\theta(i)\big)\big) * dt + x(i-1) \quad \text{(eq 1.17)}$$

$$z(i) = \big(V(i) * \cos\big(\theta(i)\big)\big) * dt + z(i-1) \quad \text{(eq 1.18)}$$

Uçuş mesafeleri, uçuş hızına eklenen mevcut hız ve açının parametresi olduğundan sırasıyla her değer için önceki değerler alınır.

2.4 İvme Denklemleri

4. mertebeden yöntem için aynı denklemler kullanılmıştır. Euler yöntemi. Bununla birlikte, denklemler adi için diferansiyel formlarına konur diferansiyel denklemler (ADD) çözücü bunları okumak ve kullanmak. Denklemlerin diferansiyel formları aşağıda listelenmiştir.

$$V(t) = \frac{d}{dt} * (x(t)) \quad \text{(eq 1.19)}$$

$$a(t) = \frac{d}{dt} * \big(V(t)\big) = \frac{d^2}{dt^2} * (x(t)) \quad \text{(eq 1.20)}$$

$$TF = m.a(t) \quad \text{(eq 1.21)}$$

$$a(t) = \frac{d}{dt} \quad \text{(eq 1.22)}$$

$$R = D + g \cdot \cos\theta \quad \text{(eq 1.23)}$$

Net Doğrusal Kuvvet= $TF - R$ olur.

$$\frac{d}{dt} * \big(V(t)\big) = \frac{TF-}{m} \Rightarrow \frac{d}{dt} * \big(V(t)\big) = \frac{TF}{m} - \frac{D}{m} - g \cdot \cos\theta \quad \text{(eq 1.24)}$$

Bu denklemler sonrası hız değerini bulmuştuk. O kısmı buradan sonra yazalım

$TF = m.a(t)$ olduğundan $a(t) = \frac{TF}{m}$ olacaktır. R sürtünmeleri çıkarsa

$$a(t) = \frac{TF}{m} - \frac{D}{m} - g \cdot \cos\theta \quad \text{(eq 1.25)}$$

Bu denklemde θ zamanla değişiyor.

θ nın denklemini de bulup $a(t)$ de $\cos\theta$ değeri buna göre bulunmalı Dikey Uçuş Açısı Newton'un 2. Yasasından elde edilirse

$$\frac{d\theta}{dt} = \frac{g}{v} * \sin(\theta) \quad \text{(eq 1.26)}$$

Türev işlemi için tekrar geri fark formülü kullanıldığında

$$\frac{\theta_i - \theta_{i-1}}{dt} = \frac{g}{v_i}\sin(\theta) \qquad \text{(eq 1.27)}$$

$$\theta_i = \theta_{i-1} + \frac{g}{v_i}\sin(\theta_{i-1})dt \qquad \text{(eq 1.28)}$$

Bunun sonucunda ivme formülümüz

$$\frac{dV}{dt} = k_t - k_d(V)^2 - g.\sin(\theta) \qquad \text{(eq 1.29)}$$

2. ATMOSFER MODEL

Dünya'nın yerden uzaya atmosferinin bir modelidir. Dünya atmosferinin yerden uzaya ampirik, küresel bir modelidir. Atmosfer bileşenlerinin sıcaklıklarını ve yoğunluklarını modeller. Bu modelin birincil kullanımı, atmosferik sürüklenme nedeniyle uydu yörünge bozulmasının tahminlerine yardımcı olmaktır. Bu model aynı zamanda gökbilimciler tarafından lazer kılavuz yıldızların lazer olmayan teleskoplar üzerindeki etkisini değerlendirmek için teleskoplar ve lazer ışınları arasındaki hava kütlesini hesaplamak için kullanılmıştır.

Mike Picone, Alan Hedin ve Doug Drob tarafından geliştirilen model, önceki MSIS-86 ve MSISE-90 modellerine dayanmaktadır, ancak gerçek uydu sürükleme verileriyle güncellenmiştir. Aynı zamanda anormal oksijeni de tahmin eder.

Bizden istenen şartlar hususunda atmosferde metrelere göre hava yoğunluğu hesabı ve grafikler aşağıda gösterilmiştir.

11000≥h (troposfer) için;

$$T = 15{,}04 - 0{,}00649 * h$$

$$p = 101{,}29 * \left[\frac{T + 273{,}1}{288.08}\right]^{5{,}256}$$

$$r = p/\left(0{,}2869 * (T + 273{,}1)\right)$$

Not: Bütün yükseklikler için r (hava yoğunluğu) formülü sabittir.

25000≥h≥11000 (alt statospher)

İçin

$$T = -56{,}46$$

$$p = 22{,}65 * e^{(1{,}73 - 0{,}000157 * h)}$$

$$r = p/\left(0{,}2869 * (T + 273{,}1)\right)$$

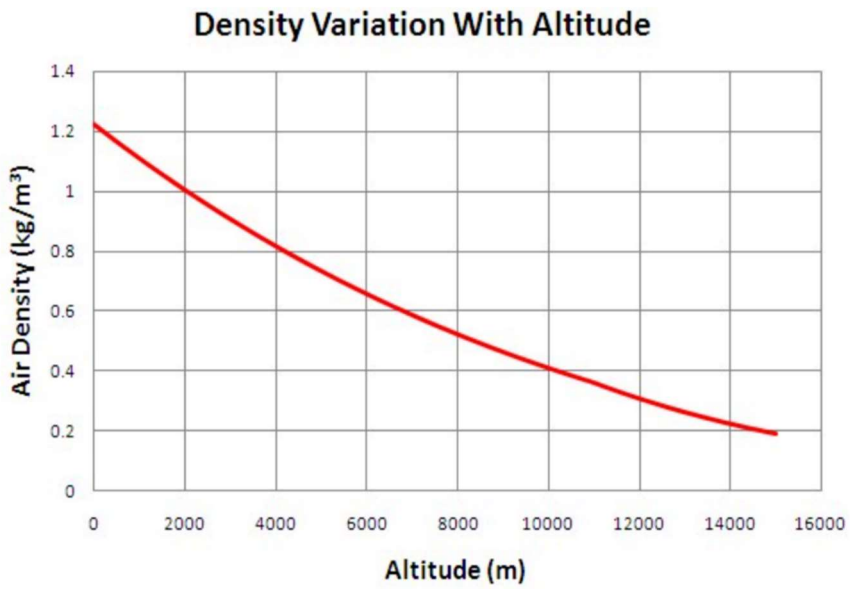
h≥25000

için

$$T = -131{,}21 + 0{,}00299 * h$$

$$p = 2{,}488 * \left[\frac{T + 273{,}1}{216{,}6}\right]^{-11{,}388}$$

$$r = p/\left(0{,}2869 * (T + 273{,}1)\right)$$

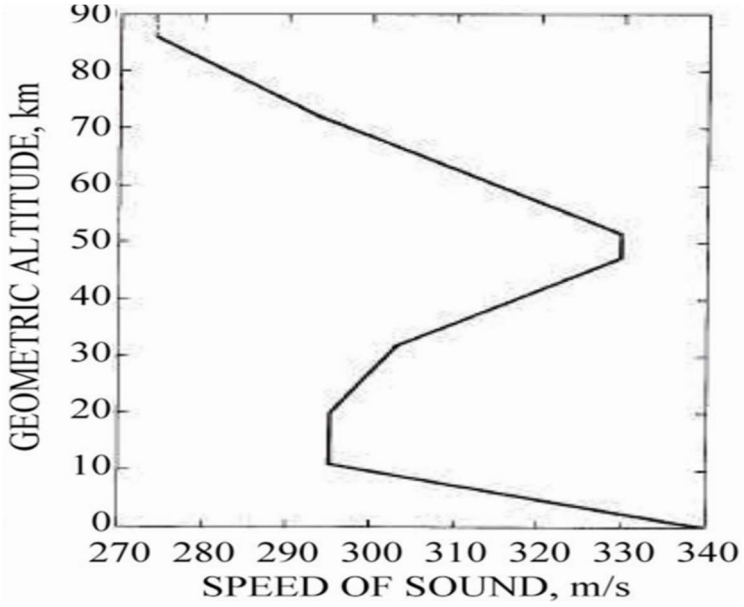


Şekil 2.1 Hava yoğunluğu – deniz seviyesi yüksekliği grafiği (0 – 16000)

Bu grafiğin buluna bilmesi için ses hızı – hava yoğunluğu formülünün bilinmesi gerekmektedir.

Aşağıda verildiği gibi;

$$V_{ses} = \sqrt{\frac{YRT}{M}}$$



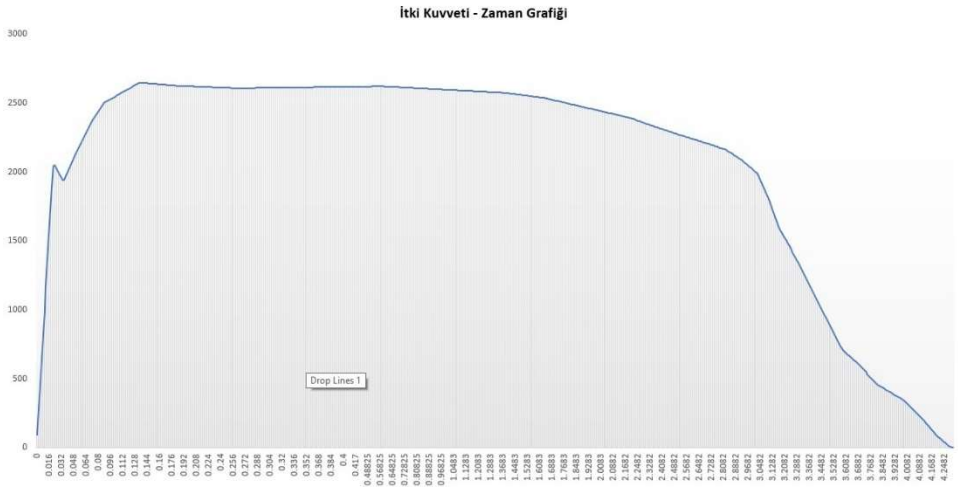
Şekil 2.2 Ses hızı – deniz seviyesi yüksekliği grafiği (0 - 90000m)

3. MOTOR MODEL

Teknofest Roket yarışması kapsamında motor modeli Cesaroni M2020 Pro75 üzerinden ele alınmıştır. Motorda yakıt tüketimi aşağıdaki formül ile hesaplanmaktadır.

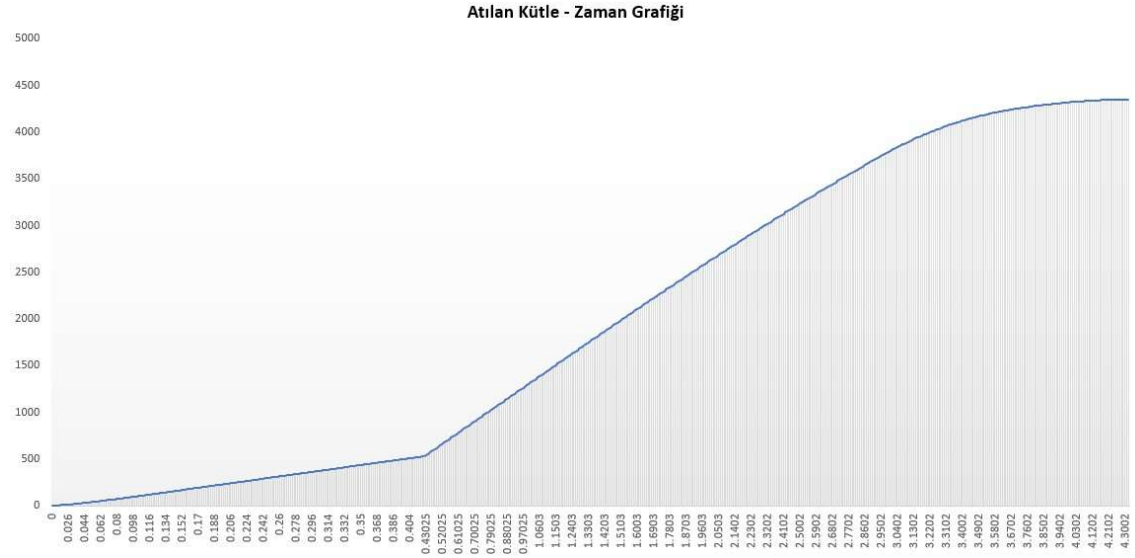
$$\dot{m}_t = \frac{F}{I_{sp}}$$

İtki – Zaman grafiği Şekil 3.1'de verilmiştir.



Şekil 3.1 İtki – Zaman Grafiği

Atılan Yakıt Kütlesi – Zaman grafiği Şekil 3.2'de verilmiştir.



Şekil 3.2 Atılan Kütle – Zaman Grafiği

4. AERODİNAMİK MODEL

Aerodinamik modelin nasıl çalıştığını anlayabilmek için aşağıdaki denklemi adım adım incelememiz gerekiyor.

$$Fx = \frac{Cd\ x\ P\ x\ V^2 x\ S}{2}$$

$$Fz = \frac{Cz\ x\ P\ x\ V^2 x\ S}{2}$$

$$M = \frac{Cm\ xP\ x\ V^2 x\ S\ x\ d}{2}$$

Girdiler:

Cd =X yönündeki sürüklenme kat sayısı

Cz = Z yönünde sürüklenme kat sayısı

Cm = Momentum kat sayısı

V = Roketin hızı

ρ =Havanın yoğunluğu

d = Roketin çapı

A = Roketin kesit alanı

q_∞:Dinamik basınç

Aerodinamik modeli ilk çıktı olarak x eksenindeki sürüklenme kuvvetini bulmak için aşağıdaki formülü inceleyeceğiz.

$$Fx = \frac{Cd\ x\ P\ x\ V^2 x\ S}{2}$$

Cd 'yi aşağıdaki formülden yararlanarak bulabiliriz.

$$Cd = \frac{D}{\frac{1}{2} x\ \rho\ x\ V^2\ x\ A}$$

İlk olarak S (referans alanı) bulmalıyız. S değerini bulmak için:

$$S = \frac{d^2\ x\ \pi}{4}$$

Yukarıda belirtilen fonksiyonda değerlerini girerek referans alanını (S) buluruz.

Elde ettiğimiz sonuçlar ile roketin x yönündeki sürüklenme kuvvetini elde etmiş oluruz. Aerodinamik modelinin ikinci çıktı olarak z eksenindeki sürüklenme kuvvetini bulmak için aşağıdaki formülü inceleyeceğiz.

$$Fz = \frac{Cz\ x\ P\ x\ V^2 x\ S}{2}$$

Cz standart girdilerde çok küçük değer aldığından dolayı ihmal edilmiştir.

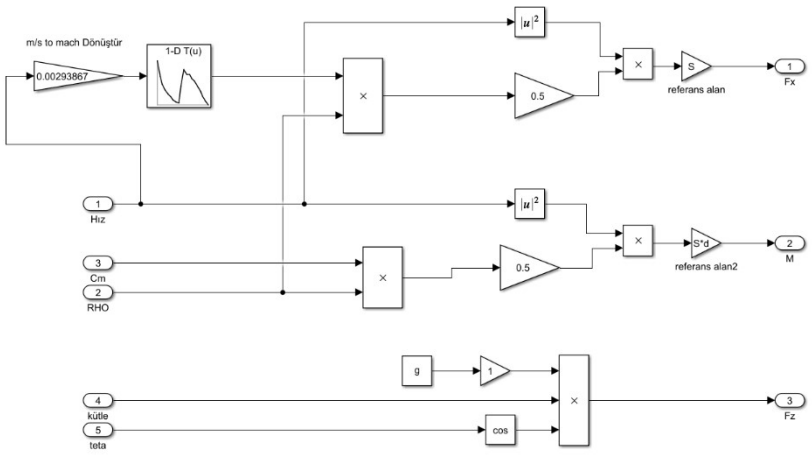
Bu formülü yaparak roketin z yönündeki sürüklenme kuvvetini elde etmiş oluruz. Aerodinamik modelinin üçüncü çıktı olarak roketin momentumunu bulmak için aşağıdaki formülü inceleyeceğiz:

$$M = \frac{Cm\ x\ P\ x\ V^2 x\ S\ x\ d}{2}$$

Cm 'yi aşağıdaki formüle göre bulabiliriz.

$$C_m = \frac{M}{q_\infty Sc}$$

Bu formül diğerlerinden farklı olarak roketin çapını (d) çarpım olarak ekleriz. Bunların sonucunda roketin momentumunu elde etmiş oluruz. Bu formüllerin sonucunda aerodinamik modelimiz aşağıdaki simulink bloğu gibidir.



Bu diyagramın sonucunda x ve z eksenindeki sürüklenme kuvvetini ve roketin momentumunu elde etmiş oluruz.

Yaptığımız modellerin 3 serbestlik olması için dinamik ve kinematik model kısmında x ve z eksenindeki hızları bulmak için roketin açısal hızını ve yaptığı teta açısını bulmamız gerekmektedir.Bundan dolayı aşağıdaki denklemi adım adım inceleyerek açısal hızını ve teta açısını elde etmiş oluruz.

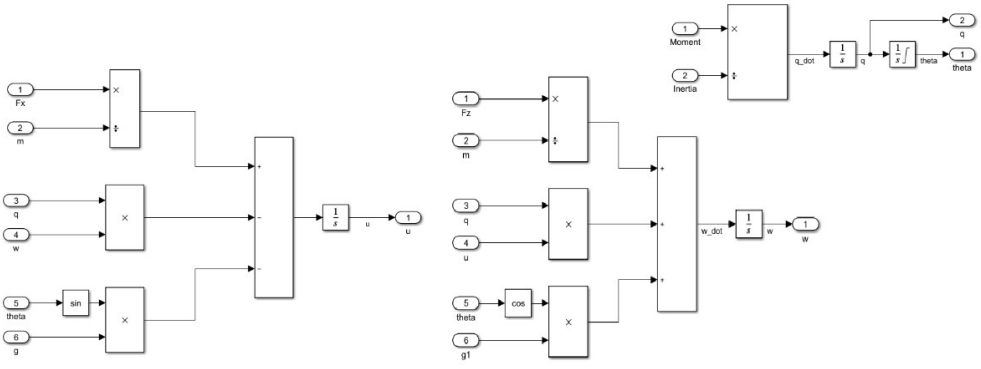
$$\dot{q} = \frac{M}{I_{YY}}$$

Bu denklemde açısal hızın zamana göre türevini elde etmiş oluruz. "$\dot{q}$"değerinin integralini aldığımız zaman bize roketin açısal hızınnı vermiş olur.

$$q = \int \dot{q}$$

Açısal hızın integralini aldığımız zaman ise teta açısını elde etmiş oluruz.

$$\theta = \int q$$



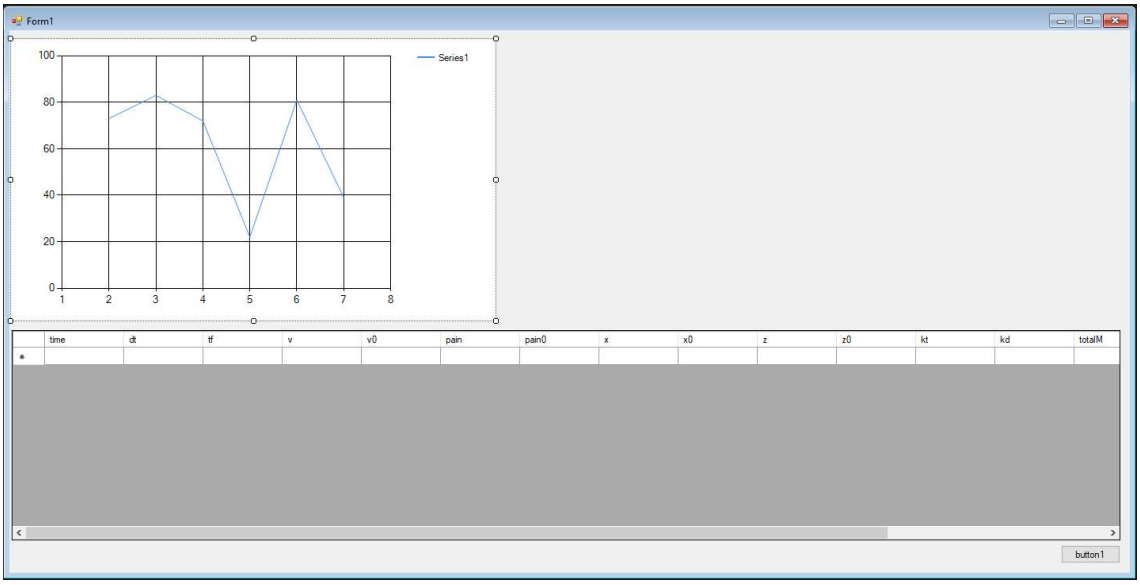
5. BENZETİM YAPISI

Uçuş benzetimi için Windows Form Application ve C# dilinden yararlanılmıştır. Benzetim 0.01 sn aralığında KTR UBR Raporu Tablo 1 ve Tablo 2'de verilen değerler ile kinematik denklemler çözümlenmiştir. Hesaplar Mach değeri üzerinden bulunmuştur.

Hazırlanan uçuş benzetimi yazılımı için tasarlanan ekran üzerinde şu bileşenler eklenmiştir;

- Chart
- DataGridView
- Label
- TextBox
- Timer

Ekran üzerinde Chart çizgi grafiğini oluşturmak için kullanılmaktadır. Timer nesnesi ile 0.01saniyede hesaplama işlemleri yapılması için kullanılmaktadır. Bu işlem döngü ile de yapılabilmekte olup yazılımımız içerisinde timer interval değerinin 10 ms olarak ayarlanması ile yapılmıştır. Ekranda sonuçların kolay görünmesi için bir tablo oluşturulmuş ve istenilen tablo şeklinde sonuç verilmesi sağlanmıştır. Uçuşun her saniyedeki konum bilgisinin de izlenebilmesi için DataGridView ekrana eklenmiştir. Ekran tasarımı Şekil 3.1'de verildiği gibidir.



Şekil 3.1 Uçuş Benzetimi Ekran Tasarımı

Ekran tasarımının tamamlanması ile kinematik denklemleri kullanılarak *Calculate* isimli bir hesaplama sınıfı oluşturulmuştur. Sınıfın içerisinde açı, hız ve konum hesaplamaları yapılmaktadır.

```
public class Calculate
    {
        public static double V(double kt, double pain, double kd, double dt,
double vi)
        {
            double response = 0.0;
            response = (kt - ConstantByTeknofestRocket.g * Math.Cos((Math.PI /
180) * pain) - kd * Math.Pow(vi, 2)) * dt + vi;
            return response;
        }

        public static double X(double v, double pain, double dt, double xi)
        {
            double response = 0.0;
            response = (v * Math.Sin((Math.PI / 180) * pain) * dt) + xi;
            return response;
        }

        public static double Z(double v, double pain, double dt, double zi)
        {
            double response = 0.0;
            response = (v * Math.Cos((Math.PI / 180) * pain) * dt) + zi;
            return response;
        }

        public static double Pain(double v, double pain, double dt)
        {
            double response = 0.0;
            response = (ConstantByTeknofestRocket.g / v * Math.Sin((Math.PI /
180) * pain) * dt + pain);
            return response;
        }

        public static double kt(double Tf, double m)
        {
            double response = 0.0;
            response = Tf / m;
            return response;
        }

        public static double kd(double p, double cd, double A, double m)
        {
            double response = 0.0;
            response = (p * cd * A) / 2 * m;
            return response;
        }

        public static double Mach(double v)
        {
            double response = 0.0;
            response = v / ConstantByTeknofestRocket.c;
            return response;
        }
    }
```

Yazılım içerisinde hesaplamalarda kullanılmak ve verilerin her 0.01 saniyedeki değerlerini tutabilmek için *DOFModel* isimli bir sınıf oluşturulmuştur. Bu sınıf içerisinde zaman, hız, konum ve açı gibi bilgiler tutulmaktadır. Burada her modelin içinde bir önceki hız, açı ve konum bilgilerinin tutulması sebebiyle hesaplama hızlandırılmıştır.

```
public class DOFModel

    {
        public double time { get; set; }
        public double dt { get; set; }
        public double tf { get; set; }
        public double v { get; set; }
        public double v0 { get; set; }
        public double pain { get; set; }
        public double pain0 { get; set; }
        public double x { get; set; }
        public double x0 { get; set; }
        public double z { get; set; }
        public double z0 { get; set; }
        public double kt { get; set; }
        public double kd { get; set; }
        public double totalM { get; set; }
        public double fuelM { get; set; }
        public double burningFuelM { get; set; }
        public double mach { get; set; }
    }
```

Zamana bağlı olarak itki kuvvetleri farklılık gösterdiği için *FModel* adında bir sınıf tanımlanmış ve form içerisinde Teknofest'in verdiği değerler ile doldurulmuştur.

```
    public class FModel
    {
        public double time { get; set; }
        public double F { get; set; }
}
```

Uçuş benzetimi için hazırlanan yazılım için farklı metodlarda da kullanıldığı için global değişkenler tanımlanmıştır. Çözüm yapılırken öncelikle zaman birimi milisaniyedir. Bu değer tMs isimli değişende tutulmakta olup formül'e uygulanırken bu tMs değişkenindeki değer 1000'e bölünerek elde edilen saniye değeri time değişkeninde tutulmaktadır. Her çözüm üresinde o anki değerler *tempModel* isimli değişken içerisinde saklanmaktadır. Her çözüm üresinde tepe noktaya gelindiği kontrol edilir. Kontrol edilme işlemi apogeeH değişkeni ile sağlanmakta olup tepe noktasına gelindiğinde o anki hız değeri apogeeV, zaman değeri apogeeT değişkeni içerisinde saklanmaktadır.

```
        public List<DOFModel> dofModel;
        public DOFModel tempModel;
```

```
        public List<FModel> fModel;
        public double total = 0.00;
        int sayac = 0;
        public BindingList<DOFModel> dofBindingList;
        BindingSource source;
```

Uygulama ilk çalıştırıldığında grafiğin aralığının çizilmesi için Chart bileşeni üzerinde düzenlemeler yapılmıştır. Grafiğin yükseklik aralığı -100 ile 600 arasında olup değer aralığı 100'er şekilde artmaktadır. Grafiğin menzil aralığı 0 ile 700 arasında olup değer aralığı 100'er artmaktadır. Anlık değişkenimizin içerisine verilen başlangıç değerleri girilmiştir. Girilen değerler ile global tanımlanmış konum, hız ve açı değerleri hesaplanmıştır. Hesaplama sonrası timer başlatılarak 0.01 saniye aralığında hesaplama yapılması sağlanmıştır.

```
double p = 0.0;
            sayac++;
            total = timer1.Interval/10000.0 * sayac;
            total = Math.Round(total,2);
            try
            {
                tempModel = new DOFModel();
                tempModel.pain0 = dofModel.Last().pain;
                tempModel.v0 = dofModel.Last().v;
                tempModel.x0 = dofModel.Last().x;
                tempModel.z0 = dofModel.Last().z;
                tempModel.time = total;
                tempModel.dt = 0.01;
                tempModel.tf = fModel.FindAll(x => x.time ==
total).FirstOrDefault() == null ? 0.0 : fModel.FindAll(x => x.time ==
total).FirstOrDefault().F;
                tempModel.burningFuelM = (fModel.FindAll(x => x.time ==
total).FirstOrDefault() == null ? 0.0 : fModel.FindAll(x => x.time ==
total).FirstOrDefault().F / ConstantByTeknofestRocket.Isp)/1000.0;
                tempModel.totalM = dofModel.Last().totalM -
tempModel.burningFuelM;
                tempModel.kt = Calculate.kt(fModel.FindAll(x => x.time ==
total).FirstOrDefault() == null ? 0.0 : fModel.FindAll(x => x.time ==
total).FirstOrDefault().F, tempModel.totalM);
                if (tempModel.z0 + ConstantByTeknofestRocket.altitude<1000.0)
                {
                    p = ConstantByTeknofestRocket.pFor0;
                }
                else if (tempModel.z0 + ConstantByTeknofestRocket.altitude <
3000.0)
                {
                    p = ConstantByTeknofestRocket.pFor1000;
                }
                else if (tempModel.z0 + ConstantByTeknofestRocket.altitude <
5000.0)
                {
                    p = ConstantByTeknofestRocket.pFor3000;
                }
                else if (tempModel.z0 + ConstantByTeknofestRocket.altitude <
10000.0)
                {
                    p = ConstantByTeknofestRocket.pFor5000;
```

```
                }
                else
                {
                    p = ConstantByTeknofestRocket.pFor10000;
                }
                tempModel.kd = (ConstantByTeknofestRocket.cD * Math.PI *
Math.Pow(ConstantByTeknofestRocket.r, 2) *  p / 2 * tempModel.totalM);
                tempModel.v = Calculate.V(tempModel.kt, tempModel.pain0,
tempModel.kd, tempModel.dt, tempModel.v0);
                tempModel.pain = Calculate.Pain(tempModel.v * 34.2,
tempModel.pain0, tempModel.dt);
                tempModel.x = Calculate.X(tempModel.v * 34.2, tempModel.pain,
tempModel.dt, tempModel.x0);
                tempModel.z = Calculate.Z(tempModel.v * 34.2, tempModel.pain,
tempModel.dt, tempModel.z0);
                tempModel.mach = Calculate.Mach(tempModel.v*34.2);
                dofModel.Add(tempModel);
                chart1.Series["Series1"].Points.AddXY(total, tempModel.v * 34.2);
            }
            catch (Exception ex)
            {
                timer1.Enabled = false;
                timer1.Stop();
                MessageBox.Show("HATA");
                throw;
            }
```

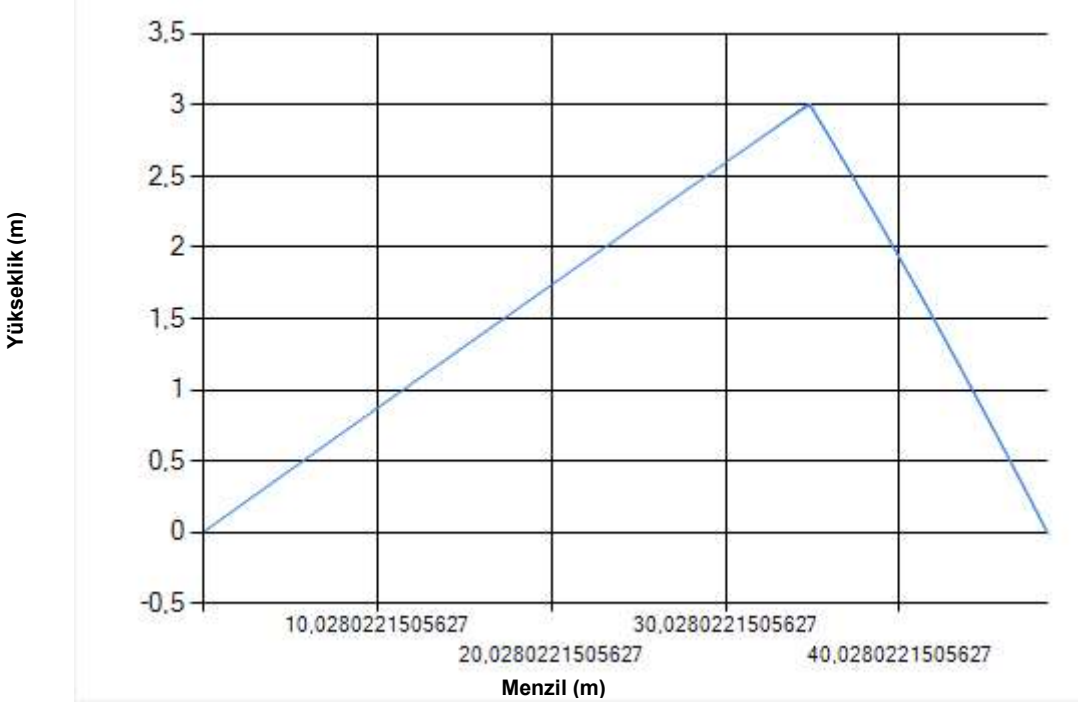
Timer bileşeni aktif edildiğinde her 0.01 saniye geçtiğinde metod içerisindeki işlemler tetiklenmektedir. Öncelikle milisaniye cinsinden zaman hesaplanır sonrasında formül çağrılmadan önce zaman saniyeye çevrilmektedir. Sonrasında önceki hareketin hızi açı ve konum bilgileri yeni modele alınmaktadır. Hız için kullanılan Kt ve Kd değerleri hesaplanarak diğer denklemlerimizin hesaplaması gerçekleştirilir. Hata alınması durumunda uçuş verilerini etkilememesi için Try- Catch hata blokları konulmuştur. Dikkat edilmesi gerekli bir diğer durum yüksekliğe göre yoğunluk bilgisinin değişmesidir. Toplam kütle yakıt kütlesi ile toplanmış her adımda atılan kütle çıkartılmıştır. İtki kuvvetleri Benzetim Doğrulaması için *Teknofest'in* verdiği değerler ile yapılmıştır. İtki kuvvetleri aşağıdaki kod bloğu ve devamı ile listede tutulmuştur.

```
fModel.Add(new FModel() { time = 0.0, F = 0.0 });
fModel.Add(new FModel() { time = 0.01 , F = 2587.180488 });
fModel.Add(new FModel() { time = 0.02 , F = 2587.180488 });
fModel.Add(new FModel() { time = 0.03 , F = 2587.180488 });
fModel.Add(new FModel() { time = 0.04 , F = 2587.180488 });
fModel.Add(new FModel() { time = 0.05 , F = 2587.180488 });
fModel.Add(new FModel() { time = 0.06 , F = 2587.180488 });
fModel.Add(new FModel() { time = 0.07 , F = 2587.180488 });
```

Tutulan veriler her adımda uçuş saniye bilgisine göre hesaplamada yakıt yanma ve hız denklemlerinde kullanılmıştır. En son durumda anlık olarak her veri ekranda grafikte ve data grid bileşeninde gösterilmiştir.

6. BENZETİMİN DOĞRULANMASI

Uçuş benzetimi için ilk konum olarak x,y,z sırasıyla 0,0,0 olarak alınmıştır. Başlangıç hızı 2 alınarak, 85 derece açılı bir eğik atış uçuşu için uçuş benzetimi yapılmıştır. Roket'e ait bilgiler UBR isterleri raporundaki Tablo 2 de verilen değerler ile analiz edilmiştir. **Yapılan analiz Mach değerleri üzerinden yapılarak elde edilmiştir.** Uçuş sonrasında kodun oluşturduğu grafik Şekil 5.1'de gösterilmiştir.



Şekil 5.1 Uçuş Benzetimi Grafik Çıktısı

Başlangıç değerleri verilen uçuş benzetimi için her 0.01 saniye aralıklarla hesaplama yapılmıştır. Uçuş tepe noktasına 5,4 saniyede ulaşmıştır. Tepe noktasının yüksekliğinin 3.006 metre olduğu görülmüştür. Uçuşun tamamlanması ile son konum x,y,z'nin sırasıyla 34.850 – 0 – 3.006 olduğu görülmüştür. Tüm sonuçlar tablo 4.1'de verildiği gibidir.

Tablo 5.1 Uçuş Benzetimi Çıktıları

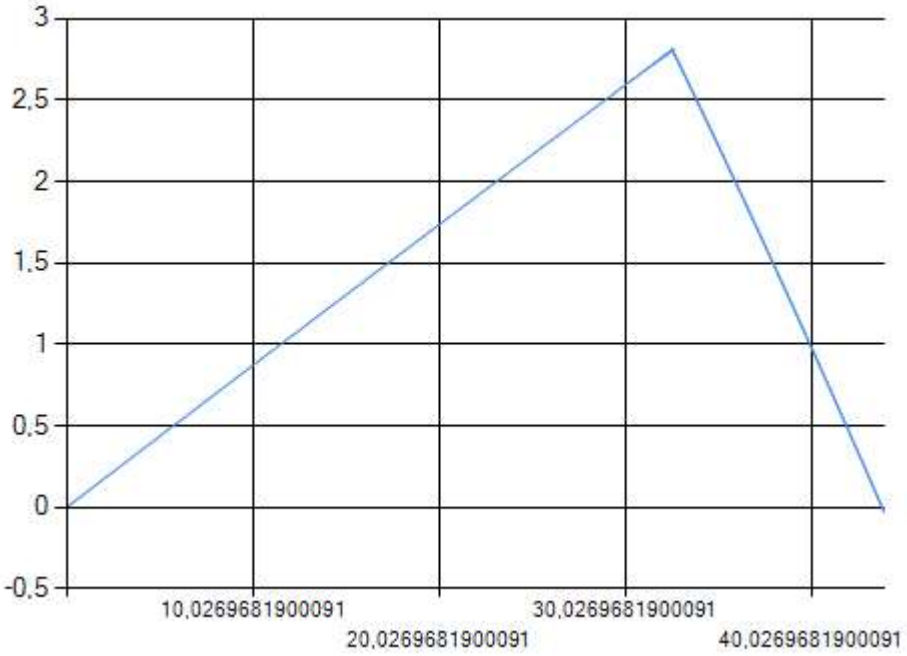
Benzetim Çıktıları	
	Değerler
Maksimum Mach Sayısı [-]	0,94393540244888485
Tepe Noktası Pozisyonu [m]	[34.850,0,3.006]
Tepe Noktası Hızı (bileşke) [m/s]	0,0012621052646317121 – (0.4286 m/s)
Tepe Noktası Mach Sayısı [-]	0,0012621052646317121
Tepe Noktası Zamanı [s]	5,4

6. BENZETİMİN DOĞRULANMASI

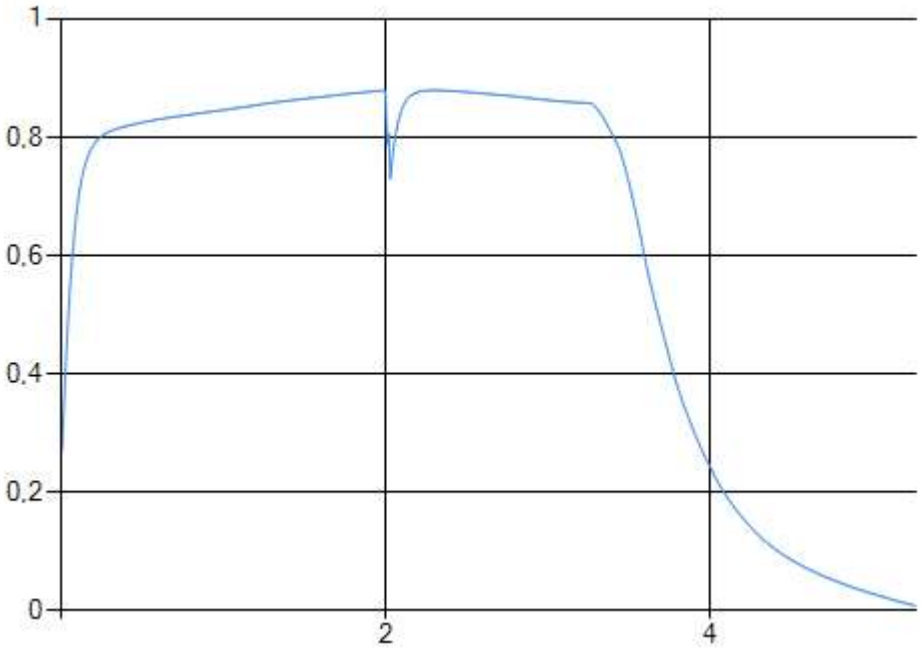
Uçuş benzetimi için ilk konum olarak x,y,z sırasıyla 0,0,0 olarak alınmıştır. Başlangıç hızı 2 alınarak, 85 derece açılı bir eğik atış uçuşu için uçuş benzetimi yapılmıştır. Roket'e ait bilgiler kendi roket bilgilerimiz kullanılarak hazırlanmıştır. Roket kütlesi 29.131, Roket çapı 0.13 alınmıştır. İtki kuvveti ve yakıt kütlelerinde kendi motor bilgilerimiz kullanılmıştır.

Tablo 6.1 Uçuş Benzetimi Çıktı ve Karşılaştırma

OpenRocket Değeri (a)		**Benzetim Değeri (b)**	**Yüzdece Fark (b-a)/a*100**
Maksimum Mach Sayısı [-]	0.76	0.88	15,78
Tepe Noktası Pozisyonu [m]	275,0,3037	[32.58, 0, 2.807,]	[-, 0, -,]
Tepe Noktası Hızı (bileşke) [m/s]	10,56	0,0029 – 9,86 m/s	-
Tepe Noktası Mach Sayısı [-]	0,030	0,0029	-
Tepe Noktası Zamanı [s]	25,5	5,35	-



Şekil 6.1 Menzil – Yörünge Grafiği



Şekil 6.2 Zaman - Mach Grafiği

REFERANSLAR

1. Anderson, C. F., and Henson, J. R., "Aerodynamic Characteristics of Several Bluff Bodies of Revolution at Mach numbers from 0.6 to 1.5," Propulsion Wind Tunnel Facility, Arnold Engineering Development Center, Air Force Systems Command, Arnold Air Force Station, Tn., AEDC-TR-71-130, July 1971.

2. Babb, C. Donald, and Fuller, Dennis E., "Static Stability Investigation of a Sounding Rocket Vehicle at Mach Numbers from 1.5 to 4.63," Langley Research Center, Langley Station, Hampton, Va., NASA TN D-4014, June 1967.

3. Briggs, Nicholas, "Multistage Supersonic Rocket Flight Simulator," Mississippi State University, Starkville, Mississippi, May 2016.

4. "Cambridge Rocketry Simulator." CambridgeRocketry:Home, Cambridge University, cambridgerocket.sourceforge.net/.

5. Ferris, James C., "Static Stability Investigation of a Single-Stage Sounding Rocket at Mach Numbers from 0.60 to 1.20," Langley Research Center, Langley Station, Hampton, Va., NASA TN D-4013, June 1967.

6. Fournier, Roger H., Hinson, William F,, and Langhans, Richard A.," Aerodynamic Characteristics In Pitch of a 1/7-Scale Model of a Two- and Three-Stage Rocket Configuration at Mach Numbers of 0.4 to 4.63," Langley Research Center, Langley Station, Hampton, Va., NASA TN D-5378, August 1969.

7. Fuller, Dennis E., "Effects of Nose Shape and Fin Geometry on Static Stability of a High-FinenessRatio Sounding Rocket," Langley Research Center, Langley Station, Hampton, Va., NASA TM X-1661, October 1968. 8. Missile Guidance and Control System [George M. Siouris]

8. Aircraft Control and Simulation [Stevens, Lewis]

9. I. V. Sorokin and A. V. Markov, "Utilization of space stations: 1971–2006," Journal of Spacecraft and Rockets, vol. 45, no. 3, pp. 600–607, 2008.

10. B. Burchett and M. Costello, "Model predictive lateral pulse jet control of an atmospheric rocket," Journal of Guidance, Control, and Dynamics, vol. 25, no. 5, pp. 860–867, 2002.

11. Z. Yongbin, "Discussion on control precision of single swing nozzle," Modern Defence Technology, vol. 35, no. 3, pp. 54–57, 2007.

12. C. F. Zhang, W. Y. Tang, H. P. Li, J. Wang, and J. C. Chen, "Application of laser tracker to thrust line measurement of solid rocket motor," Journal of Solid Rocket Technology, vol. 30, no. 6, pp. 548–551, 2007.

13. X. Ma and X. Liu, "Swing angle calibration method of SRM nozzle," Journal of Astronautic Metrology and Measurement, vol. 28, no. 5, pp. 38–42, 2008.

14. L. F. Zhang, Y. B. Guo, G. Chen, D. Ye, and R. S. Che, “Estimation of nozzle axis and vector angle by infrared photoelectric measurement technology,” Journal of Harbin Institute of Technology, vol. 42, no. 1, pp. 24–28, 201