GEBZE TEKNİK ÜNİVERSİTESİ PENÇE ROKET TAKIMI

PENÇE

TEKNOFEST-2022 ROKET YARIŞMASI
KTR AŞAMASI UÇUŞ BENZETİMİ RAPORU

İÇİNDEKİLER

KİNEMATİK VE DİNAMİK DENKLEMLER

İvme denklemi

(1) de verilen " f " kuvvet motorun uygulayacağı itkiyi , " m " roketin ağırlığını ve " a" hava sürtünmesi ile gösterilmiştir. **(1)** de verilen formülle bileşke olarak hesaplanacaktır.

$$\boldsymbol{f = m \times a} \qquad \textbf{(1)}$$

Hız denklemi

(2) de verilen denklemlerde " α " yükselme açısını , "V" başlangıç hız değerini , " t " zamanı göstermektedir. Hız vektörünün ateşleme noktası eksen takımındaki bileşenleri aşağıda verilen (2) denklemler ile hesaplanacaktır.

$$\boldsymbol{V_x = V \times cos\alpha + a * cos\alpha * t} \qquad \textbf{(2)}$$

$$\boldsymbol{V_y = V \times cos\alpha + a * sin\alpha * t}$$

Konum denklemi

(3) de verilen integral ile konum hesabı yapılacaktır. Denklemde bulunan "V_0" roketin ilk hızını, "V_1" roketin son hızını ve " x " alınan yolu temsil etmektedir.

$$\boldsymbol{\int_{V_0}^{V_1} V\, dv = x} \qquad \textbf{(3)}$$

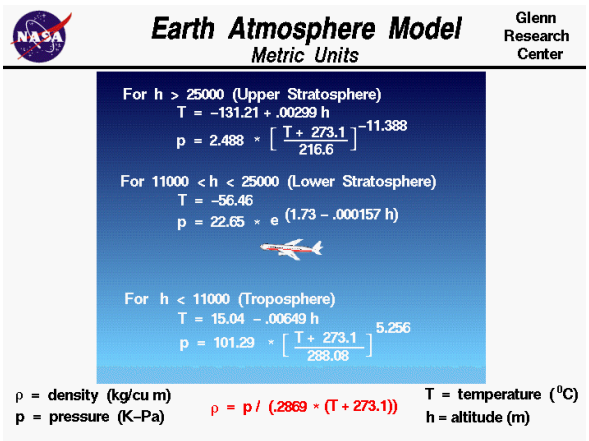
Uçuş yolu açısı hesabı denklemi

Uçuş yolu hesabı **(4)** de verilen denklem ile hızın ateşleme noktası eksen takımına göre uçuş yolu açısını hesaplamaktadır. Yükselme açısını "a" temsil etmektedir.

$$\boldsymbol{a = \arctan\left(\frac{V_y}{V_x}\right)} \qquad \textbf{(4)}$$

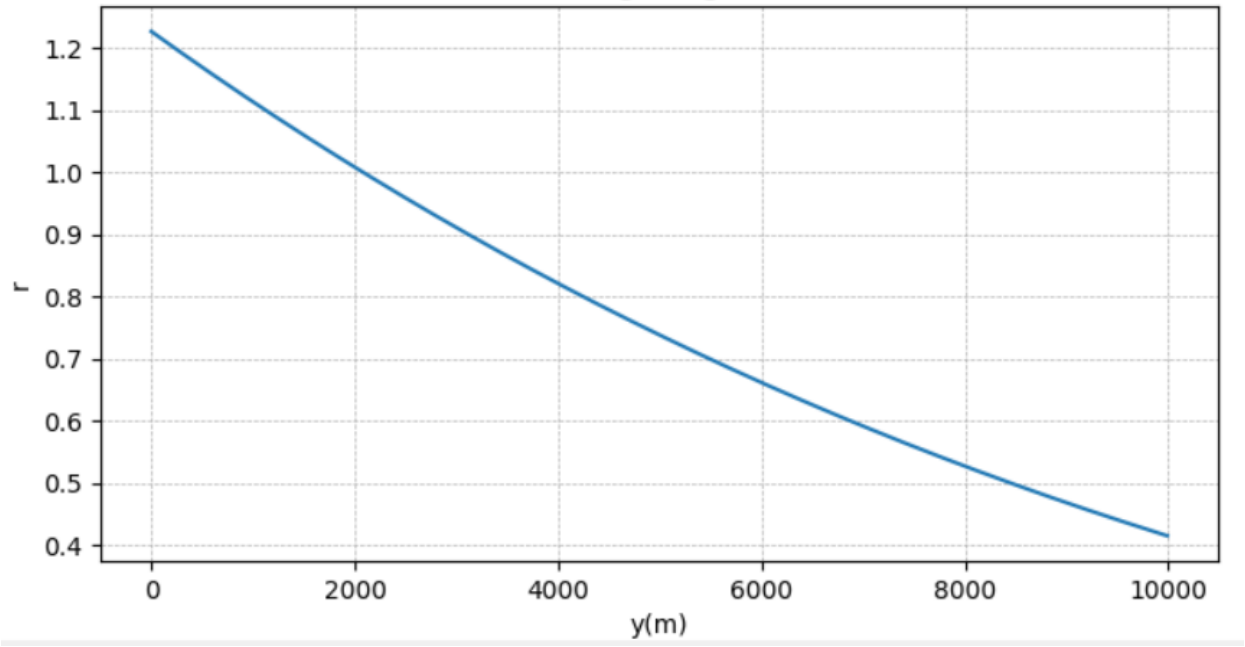
ATMOSFER MODELİ

Atmosfer modeli oluşturulurken Şekil 1'de verilen denklem setlerinden faydalanılmıştır.



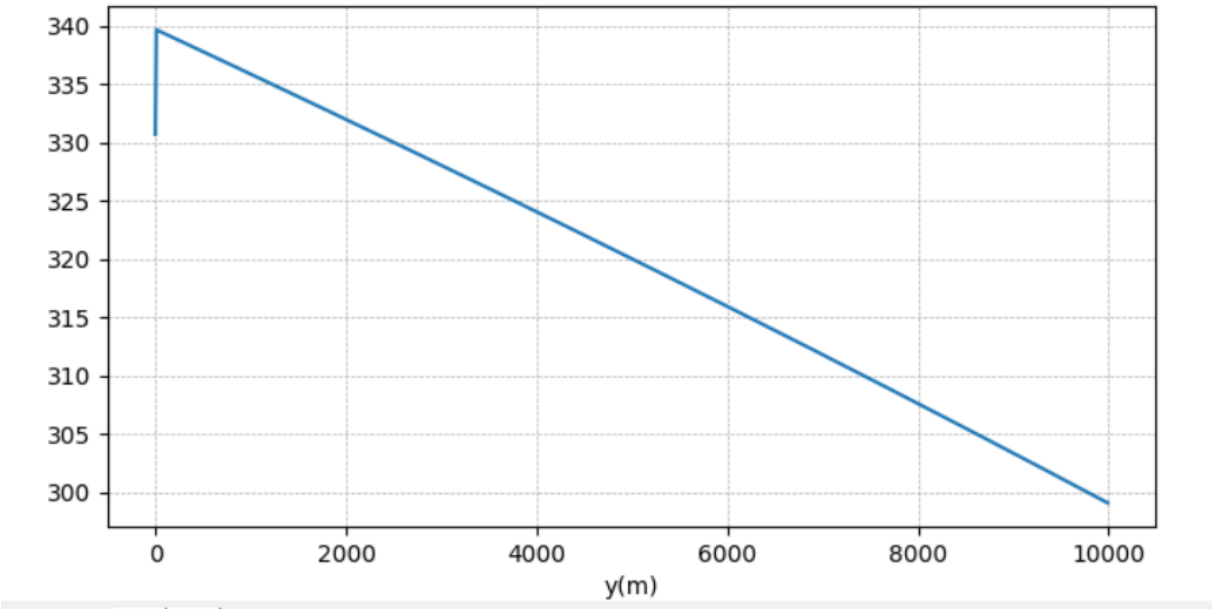
Şekil 1: Nasa Atmosfer Modeli

Şekil 2. Hava yoğunluğu – deniz seviyesi yüksekliği grafiği



Hava yoğunluğu – deniz seviyesi yüksekliği grafiği (0 m – 10000 m)

Şekil 3. Ses hızı – deniz seviyesi yüksekliği grafiği

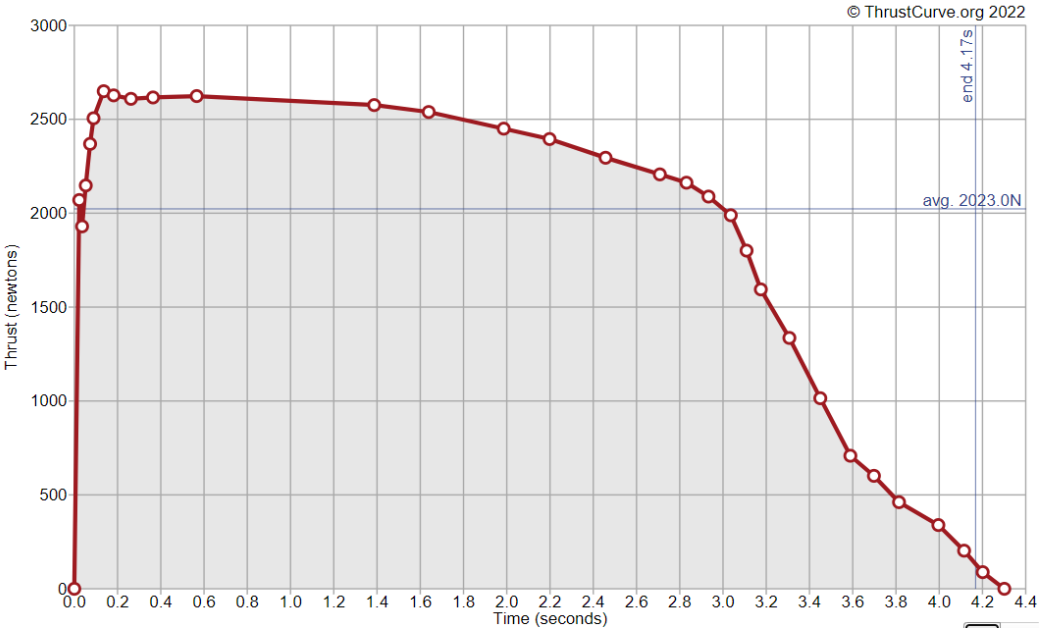


Ses hızı – deniz seviyesi yüksekliği grafiği (0 m – 10000 m)

MOTOR MODELİ

Zamana bağlı itki kuvveti modeli

Zamana bağlı itki kuvveti modeli oluşturulurken benzetim doğrulaması için paylaşılan itki verisi dosyasından faydalanılmıştır. Takımların kendi roketleri için yapacak oldukları benzetimde ise motorun itki profili dosyasından faydalanılmış olup itki profili dosyasında yer almayan veriler için doğrusal interpolasyon yapılacaktır. Şekil 4' de Cesaroni M2020 İtki profili verilmiştir.



Şekil 4. Cesaroni M2020 İtki profili

Zamana bağlı atılan kütle (harcanan yakıt kütlesi) modeli

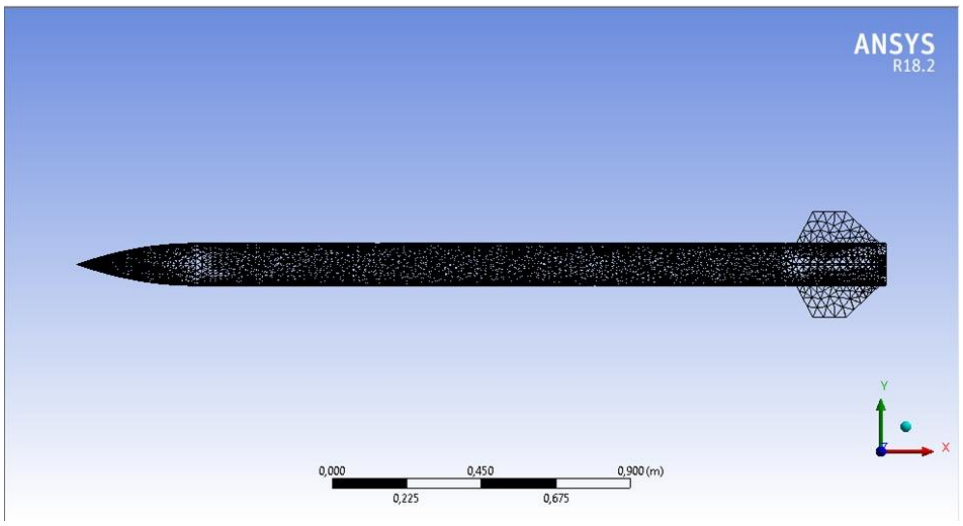
Zamana bağlı atılan kütle modeli roketin kütlesinin zamana bağlı olarak lineer bir şekilde yakıt kütlesinin harcandığı varsayımı yapılarak oluşturulmuştur. **(5)** de verilen denklem motorun anlık kütlesini hesaplarken kullanılmıştır. " m_0" motorun anlık kütlesini , "m" başlangıç yakıt kütlesini , " t " toplam yanma süresini , " Δt " anlık geçen zaman aralığını temsil etmektedir.

$$\boldsymbol{m_0 = m - \left(\frac{m}{t}\right) * \Delta t} \qquad \textbf{(5)}$$

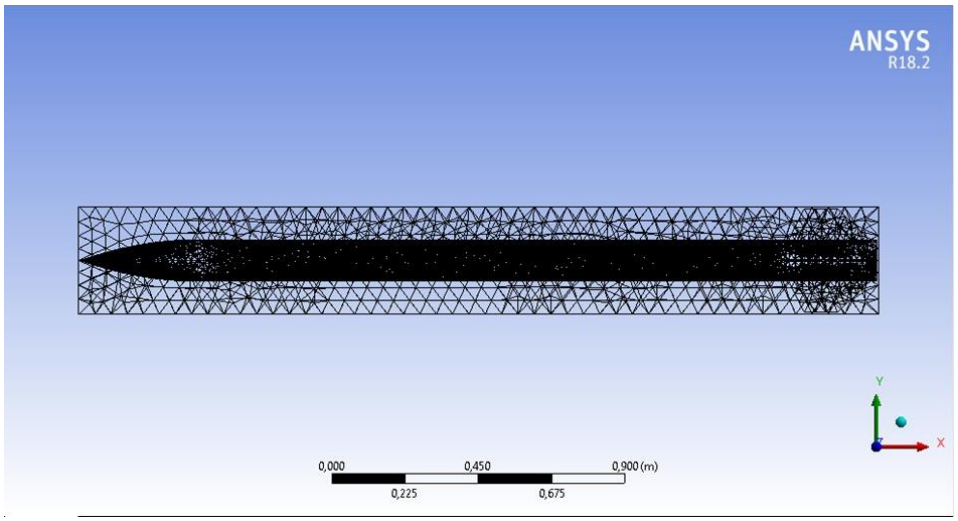
Fakat istenildiği gibi bir sonuç elde edilemedi.

Aerodinamik Model

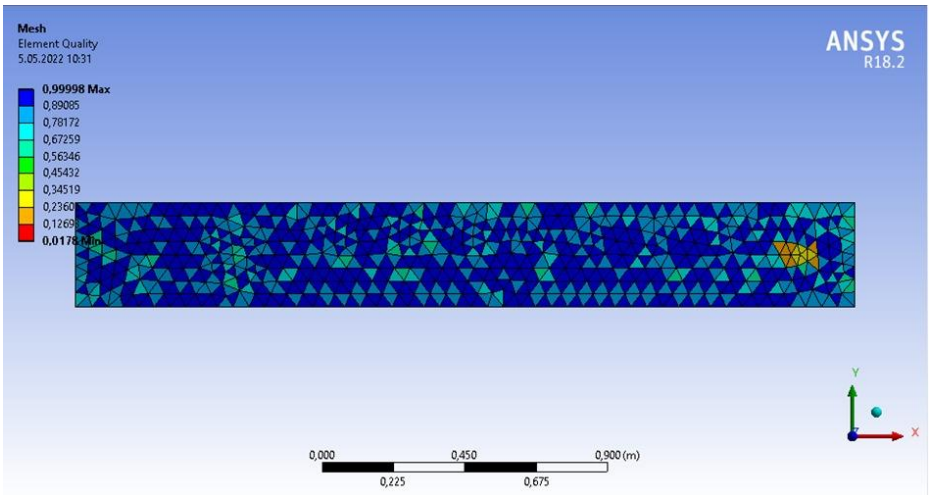
" Cd " değeri hesaplanırken ANSYS programı üzerinden yapılan " Cfd " analizinden faydalanılmıştır. Analiz sonunda elde edilen sonuçlar Şekil 5 , 6,7,8 de gösterilmiştir.



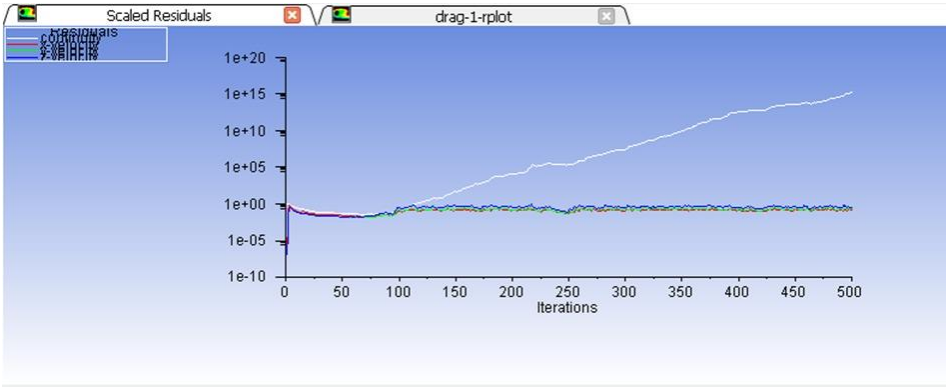
Şekil 5. ANSYS mesh işlemi body color stili çıktısı -1



Şekil 6. ANSYS mesh işlemi body color stili çıktısı -2



Şekil 7. ANSYS Mesh işlemi element quality stili çıktısı



Şekil 8. CFD Cd iterasyon grafiği

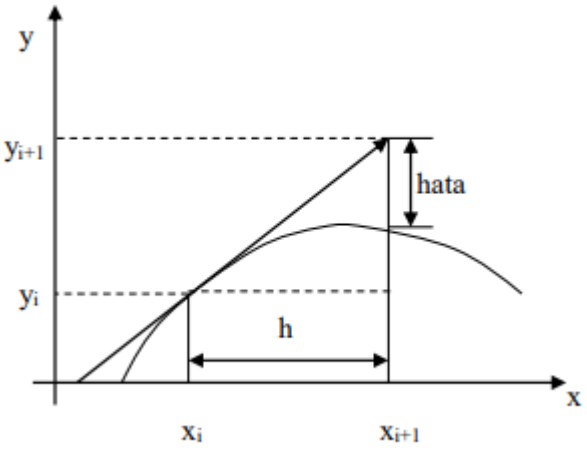
BENZETİM YAPISI

Benzetim yapılırken Nümerik Entegrayon yöntemlerinden Euler Methodu kullanılmıştır. Euler yöntemi, diferansiyel denklemlerin sayısal çözümünde kullanılan yöntemlerden birisidir. Euler yöntemi Taylor-serisinin sadece birinci dereceden terimini kullanan bir yöntemdir. **(6)** da verilen denklemler kullanılarak hesaplaması yapılır.

$$\frac{dy}{dx} = f(x,y)$$

$$y_{i+1} = y_i + f(x_i, y_i)h$$

yeni değer = eski değer + eğim * adım



Uçuş benzetimi yapılırken C# dili ile windows form uygulaması olarak uçuş benzetim programı yazılmıştır. Bu program yazılırken oluşturulan tüm nesne ve hesaplama classları **CODE** kısmında verilmiştir.

CODE

Benzetim kapsamında oluşturulan Atmosphere classı:

```
public class Atmosphere
   {
      // h=altitude
      // T=Temperature
      // P=Pressure
      //R is Gas constant
      private double R = 286;
      private double specific_heat = 1.4; //
      private double h = 0;
      private double T = 0;
      private double P = 0;
      private double density = 0;
      public double g = 9.801; //  gravitational acceleration
      private double speed_of_sound=0;

      public double H
      {
         get { return h; }
         set
         {
            if (value > 0)
            {
               h = value;
            }
            else
            {
               h = 0;
            }
         }
      }
      public double Temperature
      {
         get { return T; }
         set { T = value; }

      }
      public double Pressure
      {
         get { return P; }
         set { P = value; }
      }
      public double Density
      {
         get { return density; }
         set { density = value; }
      }
      public double Speed_of_sound
```

```
        {
            get { return speed_of_sound; }
            set { speed_of_sound = value; }
        }

        public void Calc()
        {
            speed_of_sound = Math.Sqrt(specific_heat*R*(T+273.15));
            if (h > 25000)//upper stratosphere
            {
                T = -131.21 + (0.00299 * h);
                P = 2.488 * Math.Pow(((T + 273.1) / 216.6), -11.388);
                density = P / (0.2869 * (T + 273.1));
            }
            if (h > 11000 && h < 25000)//Lover stratosphere
            {
                T = -56.46;
                P = 22.65 * Math.Pow(Math.E, (1.73 - 0.000157 * h));
                density = P / (0.2869 * (T + 273.1));
            }
            if (h < 11000)//Troposphere
            {
                T = 15.04 - 0.00649 * h;
                P = 101.29 * Math.Pow(((T + 273.1) / 288.08), 5.256);
                density = P / (0.2869 * (T + 273.1));
            }

        }
    }
```

Benzetim kapsamında oluşturulan Motor classı:

```
public class Motor
    {
        private double propellant_mass= 4.659; // kg , doğrulama için
        private double[][] thrust;

        public double Propellant_mass
        {
            get { return propellant_mass; }
        }
        public double[][] Thrust
        {
            get { return thrust; }
            set { thrust = value; }
        }
        public void MassModel(double dt)
        {
```

```
            propellant_mass -=(propellant_mass/ 4.27)*dt;
        }
    }
```

Benzetim kapsamında hesaplamalarının bir çoğunun yapılacağı Calculation classı:

```
  public class Calculations
    {

        public static double Calcualte_Mach(Rocket rocket)
        {
            rocket.Mach_number = Math.Sqrt(rocket.Velocity.x * rocket.Velocity.x +
rocket.Velocity.y * rocket.Velocity.y) / rocket.Rocket_atmospere.Speed_of_sound;
            return rocket.Mach_number;
        }
        public static double Calculate_Area(double diameter)
        {
            double area = Math.PI * diameter * diameter;
            return area;
        }
        public static Rocket Move(Rocket rocket)
        {
            rocket.Drag_force = Calculate_drag_force(rocket);


            return rocket;
        }
        public static double Calculate_drag_force( Rocket rocket)
        {
            double drag_force;
            drag_force = (rocket.Drag_coefficient * rocket.Rocket_atmospere.Density *
(rocket.Velocity.x * rocket.Velocity.x + rocket.Velocity.y *
rocket.Velocity.y)*rocket.Cross_sectionaş_area)/2;
            return drag_force;
        }




        private static double[] cdVal0 = new double[]{0.4340, 0.3954, 0.3752, 0.3617, 0.3515,
0.3429, 0.3359, 0.3309, 0.3293, 0.3700, 0.4111, 0.3996, 0.3995, 0.3908, 0.3820, 0.3696,
    0.3575, 0.3461, 0.3350, 0.3244};

        private static double[] cdVal3000 = {0.4512, 0.4099, 0.3885, 0.3741, 0.3633, 0.3542,
0.3468, 0.3415, 0.3398, 0.3805, 0.4205, 0.4083, 0.4079, 0.3990, 0.3900, 0.3773,
    0.3651, 0.3535, 0.3423, 0.3315};

        private static double[] cdVal6000 = {0.4711, 0.4267, 0.4036, 0.3882, 0.3767, 0.3670,
0.3592, 0.3536, 0.3519, 0.3924, 0.4313, 0.4181, 0.4175, 0.4083, 0.3991, 0.3862,
```

```
    0.3738, 0.3620, 0.3506, 0.3396};

        private static double[] cdVal12000 = {0.5249, 0.4715, 0.4441, 0.4259, 0.4124, 0.4012,
0.3923, 0.3858, 0.3838, 0.4240, 0.4598, 0.4443, 0.4430, 0.4331, 0.4233, 0.4098,
    0.3968, 0.3844, 0.3725, 0.3610 };

        public static double findCd(double mach, double height)
        {
            if (mach < 0.1 || mach > 2)
                mach = 0.1;

            if (Math.Ceiling(mach * 10) != Math.Floor(mach * 10)) // Virgülden sonra 1'den
fazla basamağı olan mach değerleri için lineer interpolasyon yapar
            {
                double machCeil = Math.Ceiling(mach * 10) / 10;
                double machFloor = Math.Floor(mach * 10) / 10;
                double cdCeil = findCd(machCeil, height);
                double cdFloor = findCd(machFloor, height);

                return (cdCeil - cdFloor) * (mach - machFloor) + cdFloor;
            }
            int index = (int)Math.Floor(mach * 10) - 1; // mach değerine uygun cd değeri
seçmek için mach değerini 10'la çarpıp 1 azaltarak cd array'i için uygun bir index değeri
elde eder

            if (height >= 0 && height < 3000) // mach ve height değerine karşılık gelen cd
değerini bulur ve return eder
                return (cdVal3000[index] - cdVal0[index]) * height / 3000 + cdVal0[index];
            else if (height >= 3000 && height < 6000)
                return (cdVal6000[index] - cdVal3000[index]) * (height - 3000) / 3000 +
cdVal3000[index];
            else if (height >= 6000)
                return (cdVal12000[index] - cdVal6000[index]) * (height - 6000) / 6000 +
cdVal6000[index];
            else
                return cdVal0[index];
        }


    }
```

Benzetim kapsamında oluşturulan Rocket classı:

```
    public class Rocket
    {
        private Motor motor;
        private Vector2D acceleration, velocity, thrust,position;
        private double total_mass, initial_altitude, rocket_diameter,
drag_coefficient,cross_sectionaş_area,mach_number;
```

```
        private double drag_force;
        private Atmosphere atmosphere;

        public Motor Rocket_motor
        {
            get { return motor; }
            set { motor = value; }
        }
        public Vector2D Acceleration
        {
            get { return acceleration; }
            set { acceleration = value; }
        }
        public Vector2D Velocity
        {
            get { return velocity; }
            set { velocity = value; }
        }
        public Vector2D Thrust
        {
            get { return thrust; }
            set { thrust = value; }
        }
        public Vector2D Position
        {
            get { return position; }
            set { position = value; }
        }
        public double Total_mass
        {
            get { return total_mass; }
            set { total_mass = value; }
        }
        public double İnitial_altitude
        {
            get { return initial_altitude; }
            set { initial_altitude = value; }
        }
        public double Rocket_diameter
        {
            get { return rocket_diameter; }
            set { rocket_diameter = value; }
        }
        public double Drag_force
        {
            get { return drag_force; }
            set { Drag_force = value; }

        }
        public double Drag_coefficient
```

```
    {
        get { return drag_coefficient; }
        set { Drag_coefficient = value; }

    }
    public double Cross_sectionaş_area
    {
        get { return cross_sectionaş_area; }
        set { cross_sectionaş_area = value; }
    }
    public double Mach_number
    {
        get { return mach_number; }
        set { mach_number= value; }
    }
    public Atmosphere Rocket_atmospere
    {
        get { return atmosphere; }
        set { atmosphere = value; }
    }
```

BENZETİMİN DOĞRULANMASI

Tablo 1' de Doğrulama Çalışması Koşul değerleri verilmiştir. Tablo 2'de ise Doğrulama Çalışması Diğer verileri verilmiştir.

Tablo 1. Doğrulama Çalışması Başlangıç Koşul Değerleri

	Değer
Pozisyon [m]	[0, 0, 0]
Hız (bileşke) [m/s]	2
Uçuş Yolu Açısı [derece]	85

Tablo 2. Doğrulama Çalışması Diğer Verileri

	Değer
Başlangıç Kütlesi [kg]	25
Atış Noktası Rakımı [m]	980
Başlangıç Yakıt Kütlesi [kg]	4.659
Özgül İtki (Isp) [s]	209.5
İtki Profili Dosyası	"veri_itki_F_2022.xlsx"
Aerodinamik Veri Seti Dosyası	"veri_aero_Cd_2022.xlsx"
Roket Çapı [m]	0.14

BENZETİM SONUÇLARI

Tablo 3. Standart girdi değerleri için elde eddilen çıktılar.

	Değer
Maksimum Mach Sayısı [-]	**2.57**
Tepe Noktası Pozisyonu [m]	**23472 m**
Tepe Noktası Hızı (bileşke) [m/s]	**0 m/s**
Tepe Noktası Mach Sayısı [-]	**0**
Tepe Noktası Zamanı [s]	**66,2 s**

Tablo 4. Tasarlanmış olan Roket için elde edilen çıktılar

	OpenRocket Değeri (a)	Benzetim Değeri (b)	Yüzdece Fark (b-a)/a*100
Maksimum Mach Sayısı [-]	**0,73**	**0,73**	**% 0**
Tepe Noktası Pozisyonu [m]	**3013 m**	**3061,4 m**	**% 1,6**
Tepe Noktası Hızı (bileşke) [m/s]	**0 m/s**	**0 m/s**	**% 0**
Tepe Noktası Mach Sayısı [-]	**0**	**0**	**0**
Tepe Noktası Zamanı [s]	**25,5 s**	**25,2 s**	**%0,06**

REFERANSLAR

[1] Şekil 1: Nasa Atmosfer Modeli

URL:https://www.grc.nasa.gov/www/k-12/airplane/atmosmet.html , Accesed: 04.05.2022

[2] Ses hızı hesaplamak için kullanılan denklem seti;

URL: https://www.grc.nasa.gov/www/k-12/VirtualAero/BottleRocket/airplane/sound.html. Accesed 05.05.2022

[3] Sürüklenme kuvveti hesabı

URL:https://www.grc.nasa.gov/www/k-12/rocket/drageq.html; Accesed: 04.05.2022

[4] Euler yöntemi açıklaması

URL:https://web.itu.edu.tr/yukselen/HM504/04-%20Adi%20diferansiyel%20denklemler.pdf. Accesed 05.05.2022

FORMAT İLE İLGİLİ NOT:
• **Rapor "İçindekiler" ve "Referanslar" kısımlarını içermelidir.** • **Rapor bir kapak sayfası içermelidir.** • **Yazı tipi: Times New Roman, Punto: 12, Satır Aralıkları: 1.5** • **Tüm metinler iki tarafa yaslı şekilde olması gerekmektedir.** • **Sayfa düzeni A4 olmalıdır.** • **Rapor 25 sayfayı geçmemelidir.** • **Rapor PDF formatında olmalıdır.**