

# Моделирование движения спускаемого аппарата в атмосфере

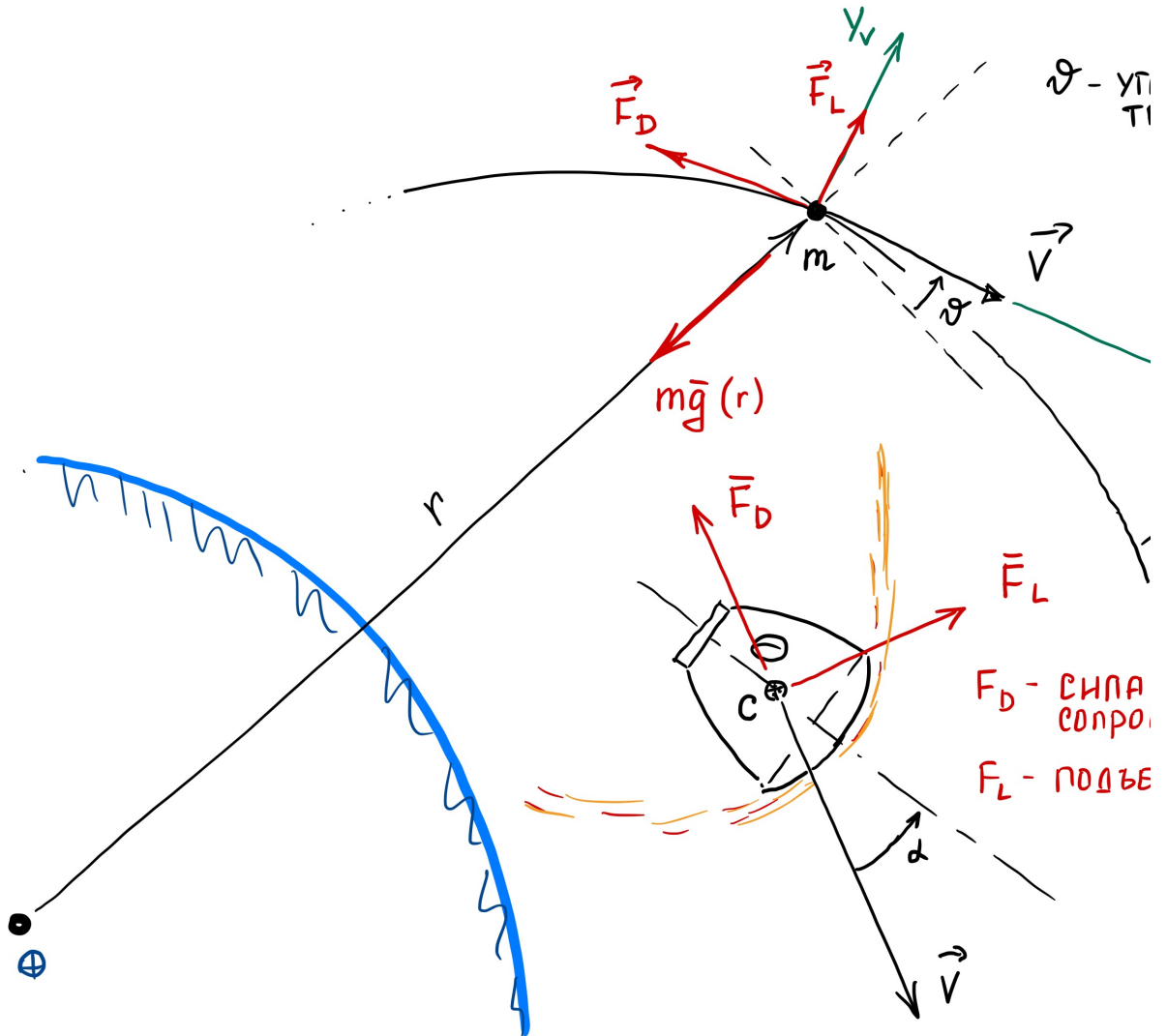
Интегрированные математические  
пакеты

Самарский университет. Кафедра теоретической механики

Юдинцев В. В.

## Модель

Рассматривается движение спускаемого аппарата в атмосфере Земли.



## Условия

- Вращение Земли не учитывается
- Спускаемый аппарат рассматривается как материальная точка постоянной массы
- Рассматривается движение спускаемого в одной плоскости
- Движение спускаемого аппарата происходит под действием силы тяжести и сил аэродинамического сопротивления

## Аэродинамические силы

На спускаемый аппарат при движении в атмосфере действуют аэродинамические силы и моменты.

Аэродинамические силы и моменты зависят от следующих параметров:

- угла атаки спускаемого аппарата;
- плотности воздуха на высоте полета;
- размеров спускаемого аппарата;
- формы спускаемого аппарата.

## Аэродинамические силы

### Представление аэродинамических сил

$$F_D = C_D (M, \alpha) \times S \times \frac{\rho V^2}{2}$$

$$F_L = C_L (M, \alpha) \times S \times \frac{\rho V^2}{2}$$

$S$  - площадь Миделя (характерная площадь) спускаемого аппарата

$V$  - скорость набегающего потока воздуха

$\rho$  - плотность воздуха (зависит от высоты полёта)

$M$  - число Маха: отношение скорости набегающего потока воздуха к местной скорости звука

$C_D$  - коэффициент лобового сопротивления, зависящий от  $M$ ,  $\alpha$ , условий обтекания

$C_L$  - коэффициент подъемной силы сопротивления, зависящий от  $M$ ,  $\alpha$ , условий обтекания

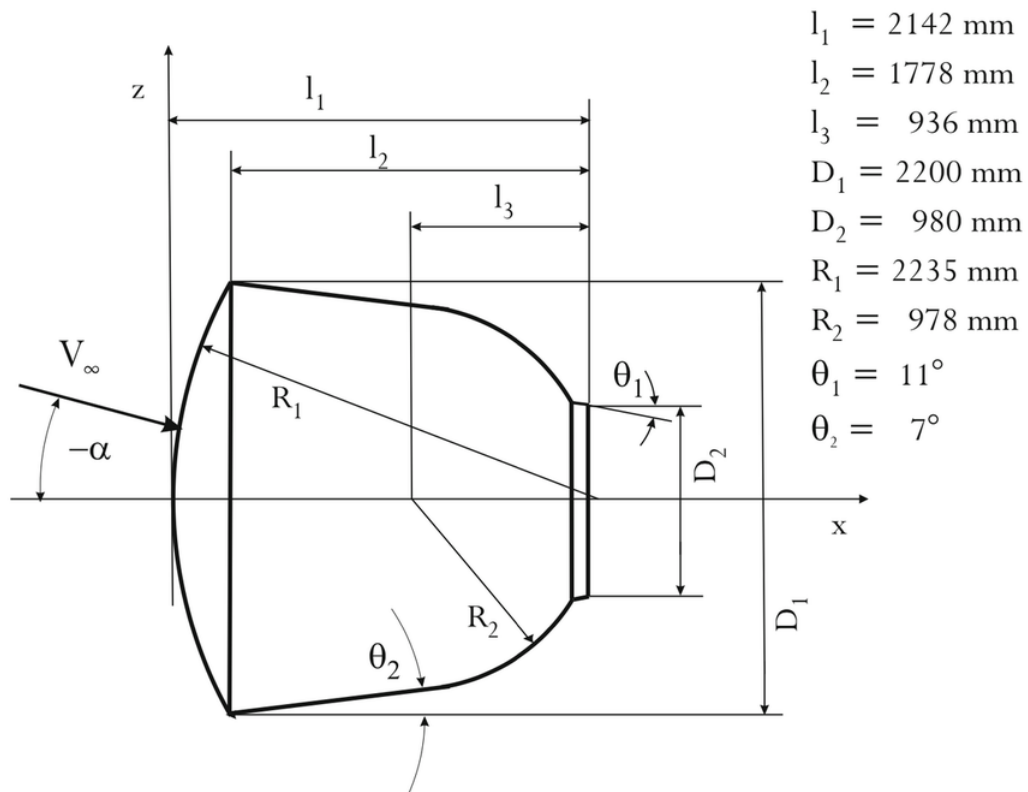
Скоростной напор (имеет размерность давления)

$$q = \frac{\rho V^2}{2} \text{ [Па]}$$

## Аэродинамические характеристики

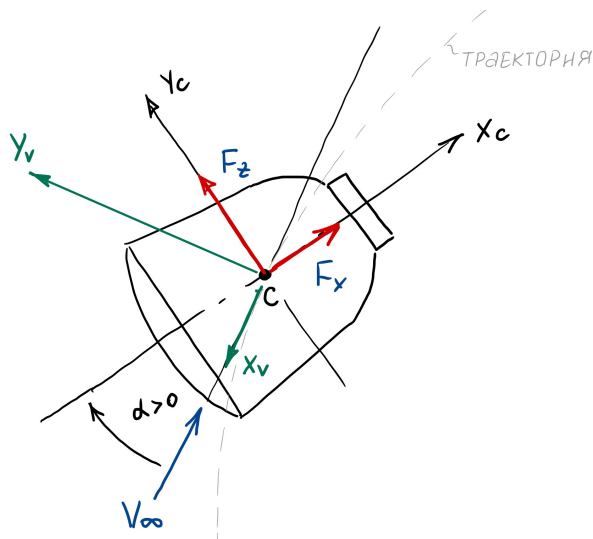
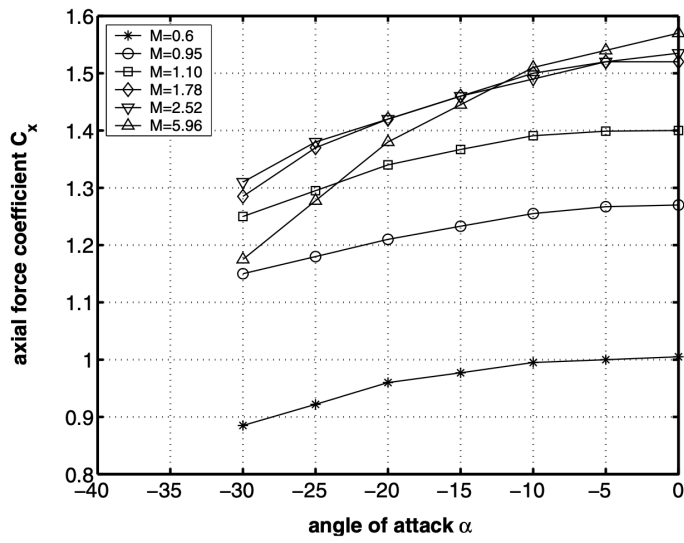
### Спускаемый аппарат космического корабля “Союз”

Масса: 3000 кг



Claus Weiland  
 Aerodynamic Data of Space Vehicles  
 Springer Heidelberg New York Dordrecht London, 2014.

# Аэродинамические характеристики



## Плотность воздуха

Зависимость плотности воздуха от высоты может быть определена при помощи термодинамической модели атмосферы, экспериментальных данных.

```
In[ ]:= StandardAtmosphereData[Quantity[10 000, "Feet"],
  "Density", Method -> "USStandardAtmosphere"]
  % // QuantityMagnitude
```

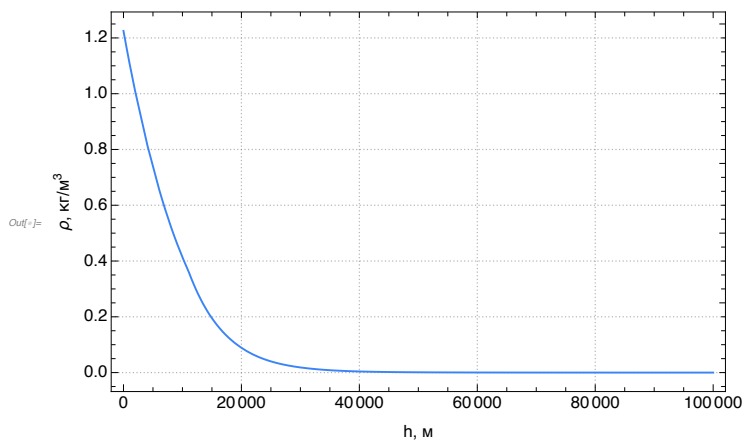
```
Out[ ]:= 0.9048484 kg/m3
```

```
Out[ ]:= 0.9048484
```

```
In[ ]:= ρ = .;
```

```
ρ[h_] := QuantityMagnitude[StandardAtmosphereData[
  Quantity[h, "Meters"], "Density", Method -> "USStandardAtmosphere"]];
```

```
In[ ]:= Plot[ρ[h], {h, 0, 100 000}, PlotRange -> All, PlotTheme -> {"Presentation", "FrameGrid"},
  FrameLabel -> {"h, м", "ρ, кг/м3"}, ImageSize -> Large]
```



```
In[ ]:= Quantity[a, "Meters"] * Quantity[b, "Newtons"]
```

```
Out[ ]:= a b m N
```



## Загрузка таблицы из файла

```
In[ ]:= SetDirectory[NotebookDirectory[]];
```


```
In[ ]:= data = Import["atm.txt"];
```

```
In[ ]:= data = Import["atm.txt", "table"];
```

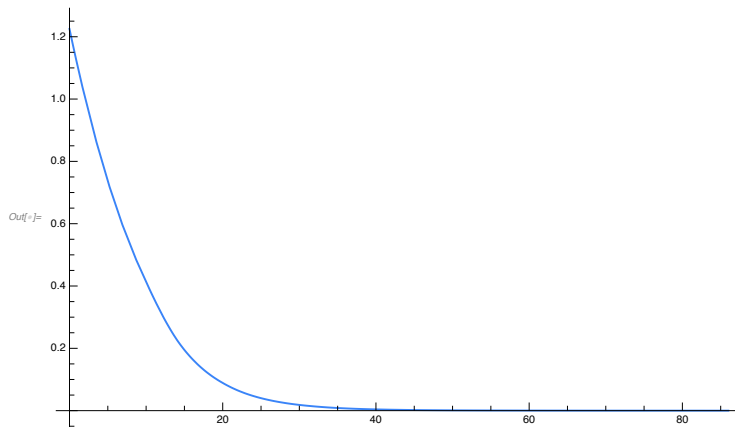
```
In[ ]:= (data[[3 ;;, {1, 7}]])
```

```
Out[ ]:= {{-2, 1.478}, {0, 1.225}, {2, 1.007}, {4, 0.8193}, {6, 0.6601}, {8, 0.5258}, {10, 0.4135},
  {12, 0.3119}, {14, 0.2279}, {16, 0.1665}, {18, 0.1216}, {20, 0.08891}, {22, 0.06451},
  {24, 0.04694}, {26, 0.03426}, {28, 0.02508}, {30, 0.01841}, {32, 0.01355},
  {34, 0.009887}, {36, 0.007257}, {38, 0.005366}, {40, 0.003995}, {42, 0.002995},
  {44, 0.002259}, {46, 0.001714}, {48, 0.001317}, {50, 0.001027}, {52, 0.0008055},
  {54, 0.0006389}, {56, 0.0005044}, {58, 0.0003962}, {60, 0.0003096}, {62, 0.0002407},
  {64, 0.000186}, {66, 0.0001429}, {68, 0.0001091}, {70, 0.00008281}, {72, 0.00006236},
  {74, 0.00004637}, {76, 0.0000343}, {78, 0.00002523}, {80, 0.00001845},
  {82, 0.00001341}, {84, 9.69 × 10-6}, {86, 6.955 × 10-6}}
```

```
In[ ]:= ρ = Interpolation[%]
```

```
Out[ ]:= InterpolatingFunction[ Domain: {{-2., 86.}}  
Output: scalar]
```

```
In[ ]:= Plot[ρ[h], {h, 0, 86}, PlotRange -> All]
```



Исправить (самостоятельно)

- Таблица слишком "короткая" (до 86 км)
- Высота в километрах. В уравнениях движения высота в метрах.

```
In[ ]:= SetOptions[Plot, PlotTheme -> {"Presentation", "FrameGrid"}, ImageSize -> Large];
```

```
SetOptions[ParametricPlot,
```

```
PlotTheme -> {"Presentation", "FrameGrid"}, ImageSize -> Large];
```

```
SetOptions[ListPlot, PlotTheme -> {"Presentation", "FrameGrid"}, ImageSize -> Large];
```

## Уравнения движения

Уравнения движения

$$\mathbf{q} = \{V[t], r[t], \theta[t], s[t]\};$$

$$\mathbf{eq} = \{$$

$$V'[t] == -g \sin[\theta[t]] - c_d S \frac{\rho[r[t] - Rz] V[t]^2}{2 m},$$

$$V[t] \times \theta'[t] == g \left( \frac{V[t]^2}{r[t] g} - 1 \right) \cos[\theta[t]] + c_L S \frac{\rho[r[t] - Rz] V[t]^2}{2 m},$$

$$s'[t] == \frac{V[t]}{r[t]} \cos[\theta[t]] Rz,$$

$$r'[t] == V[t] \sin[\theta[t]]$$

$$\};$$

Вывод уравнений приведен в книге:

В. А. Ярошевский **Вход в атмосферу космических летательных аппаратов**

М.: Наука, 1988, стр. 24.

## Параметры системы

`In[ ]:= params =`

$$\left\{ c_d \rightarrow 1.5, c_L \rightarrow 0, S \rightarrow \frac{\pi 2.2^2}{4}, m \rightarrow 3000.0, g \rightarrow 9.807, R_z \rightarrow 6371000.0, h_0 \rightarrow 100000.0 \right\}$$

`Out[ ]:= {c_d → 1.5, c_L → 0, S → 3.801327, m → 3000., g → 9.807, R_z → 6.371 × 106, h_0 → 100000.}`

`In[ ]:= nu = {V[0] == 7000, θ[0] == 0, r[0] == R_z + h_0, s[0] == 0};`

Круговая скорость на высоте h<sub>0</sub>

$$\text{In[ ]:= } \sqrt{\frac{\mu}{(R_z + h_0)}} \quad // . \mu \rightarrow 398600.4415 \times 10^9 \quad // . \text{ params}$$

`Out[ ]:= 7848.437`

`In[ ]:= params = {c_d → 1.3, c_L → 0.3, S → \frac{\pi 2.2^2}{4}, m → 3000.0, g → 9.807, R_z → 6371000.0, h_0 → 100000.0, \mu → 398600.4415 \times 10^9}`

`nu = {V[0] == \sqrt{\mu / (R_z + h_0)}, \theta[0] == 0, r[0] == R_z + h_0, s[0] == 0};`

`Out[ ]:= {c_d → 1.3, c_L → 0.3, S → 3.801327, m → 3000., g → 9.807, R_z → 6.371 × 106, h_0 → 100000., \mu → 3.986004 × 1014}`

Ускорение свободного падения зависит от высоты полета.

Например, на высоте h<sub>0</sub>:

$$\text{In[ ]:= } \frac{\mu}{(R_z + h_0)^2} \quad // . \text{ params}$$

`Out[ ]:= 9.51908`

`In[ ]:= \rho[100000]`

`Out[ ]:= 5.604 × 10-7`

## Результаты интегрирования

Проинтегрируем уравнения движения до 450 секунды полета

```
tk = 1040;
```

```
sol = NDSolve[{eq, nu} /. params, q, {t, 0, tk}]
```

```
Out[ ]:= { {V[t] → InterpolatingFunction[  

  Domain: {{0., 1.04 × 103}}  

  Output: scalar  

  ] [t],  

  r[t] → InterpolatingFunction[  

  Domain: {{0., 1.04 × 103}}  

  Output: scalar  

  ] [t],  

  θ[t] → InterpolatingFunction[  

  Domain: {{0., 1.04 × 103}}  

  Output: scalar  

  ] [t],  

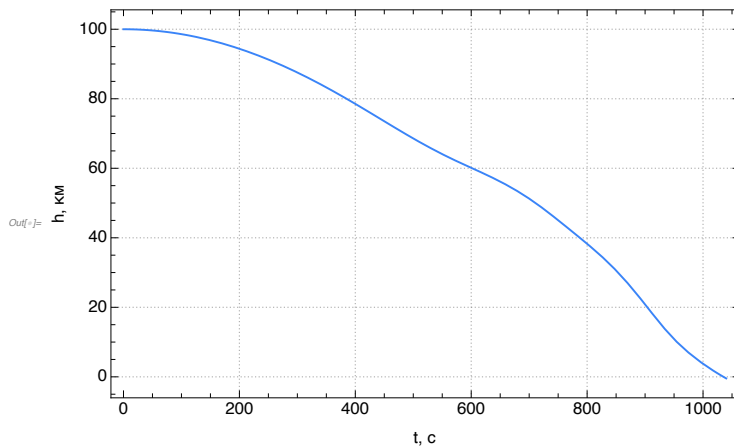
  s[t] → InterpolatingFunction[  

  Domain: {{0., 1.04 × 103}}  

  Output: scalar  

  ] [t] } }
```

```
Plot[(r[t] - Rz) * 0.001 /. params /. sol, {t, 0, tk}, FrameLabel → {"t, c", "h, км"}]
```



Конечная высота

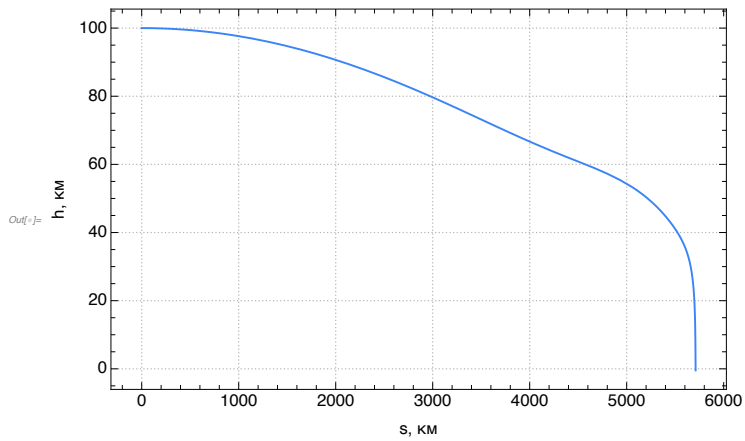
```
(r[t] - Rz) * 0.001 /. params /. sol /. t → tk
```

```
Out[ ]:= {-0.4571043}
```

## Результаты интегрирования

Траектория от дистанции полета (по поверхности Земли)

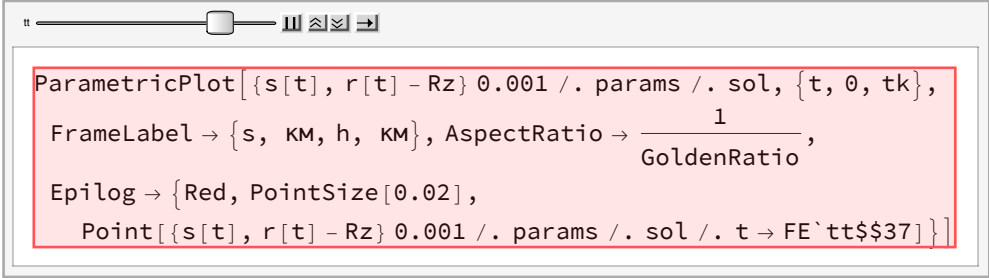
```
Out[4]= ParametricPlot[{s[t], (r[t] - Rz) * 0.001 /. params /. sol, {t, 0, tk}},  
  FrameLabel -> {"s, км", "h, км"}, AspectRatio -> 1/GoldenRatio, PlotRange -> All]
```



## Результаты интегрирования

Траектория от дистанции полета (по поверхности Земли)

```
in[ ]:= Animate[ParametricPlot[{s[t], (r[t] - Rz)} * 0.001 /. params /. sol,
  {t, 0, tk}, FrameLabel -> {"s, км", "h, км"},
  AspectRatio -> 1/GoldenRatio, Epilog -> {Red, PointSize[0.02],
  Point[{s[t], (r[t] - Rz)} * 0.001 /. params /. sol /. t -> tt]}], {tt, 0, tk}]
```

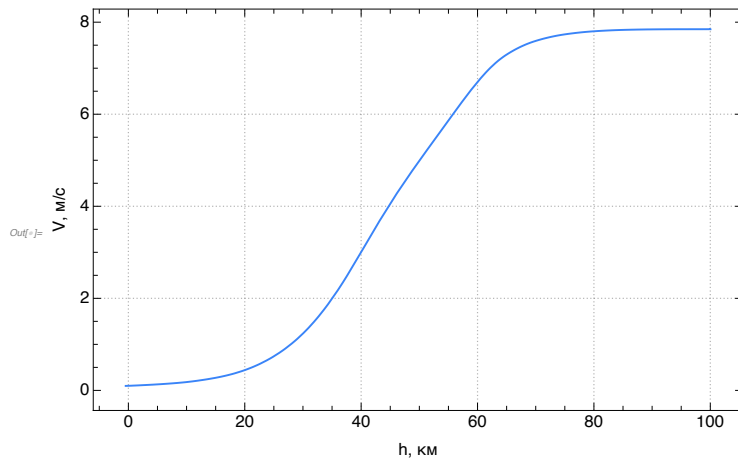


```
out[ ]:= ParametricPlot[{s[t], r[t] - Rz} 0.001 /. params /. sol, {t, 0, tk},
  FrameLabel -> {s, км, h, км}, AspectRatio ->  $\frac{1}{\text{GoldenRatio}}$ ,
  Epilog -> {Red, PointSize[0.02],
  Point[{s[t], r[t] - Rz} 0.001 /. params /. sol /. t -> FE`tt$$37]}]
```

## Результаты интегрирования

СКОРОСТЬ ОТ ВЫСОТЫ

```
Plot[ParametricPlot[{(r[t] - Rz), V[t]} * 0.001 /. params /. sol, {t, 0, tk},  
FrameLabel -> {"h, км", "V, м/с"}, AspectRatio -> 1/GoldenRatio]
```



## Управление процессом интегрирования

Остановка процесса численного интегрирования при достижении нулевой высоты.

Функция **WhenEvent**

```
In[ ]:= tk = 1100;
sol = NDSolve[
  {
    eq, nu, WhenEvent[r[t] < Rz /. params, "StopIntegration"]
  } /. params, q, {t, 0, tk}
]
```

**NDSolve**: Event location failed to converge to the requested accuracy or precision within 100 iterations between t = 1035.196532461057 and t = 1035.2458548531633.

```
Out[ ]:= { {V[t] -> InterpolatingFunction[
  Domain: {{0., 1040.}}
  Output: scalar
] [t],
  r[t] -> InterpolatingFunction[
  Domain: {{0., 1040.}}
  Output: scalar
] [t],
  ̸[t] -> InterpolatingFunction[
  Domain: {{0., 1040.}}
  Output: scalar
] [t],
  s[t] -> InterpolatingFunction[
  Domain: {{0., 1040.}}
  Output: scalar
] [t] } }
```

```
In[ ]:= tk = sol[[1, 1, 2, 0, 1, 1, 2]]
```

```
Out[ ]:= 1035.222
```

Конечная высота

```
In[ ]:= r[t] - Rz /. sol /. params /. t -> tk
```

```
Out[ ]:= { -9.126961 × 10-8 }
```







## Управление процессом интегрирования

Остановка процесса численного интегрирования при достижении нулевой высоты.

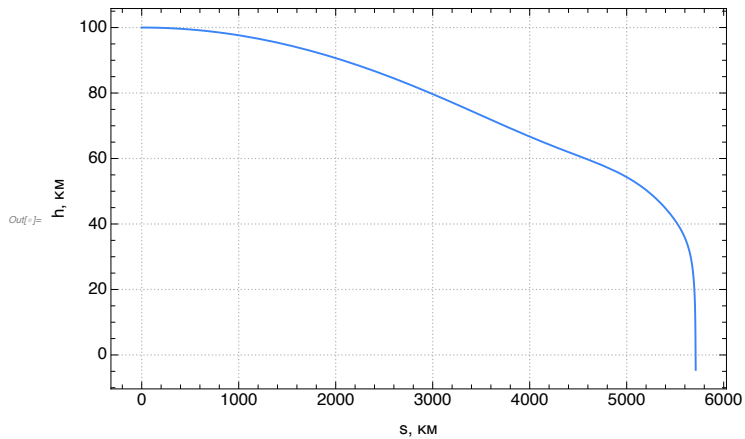
Функция **WhenEvent**

```

In[ ]:= conditions = r[t] < Rz + # & /@ Range[0, 50 000, 10 000]
Out[ ]:= {r[t] < Rz, r[t] < 10 000 + Rz, r[t] < 20 000 + Rz, r[t] < 30 000 + Rz, r[t] < 40 000 + Rz,
r[t] < 50 000 + Rz}
In[ ]:= tk = 1100;
{sol, velocities} = Reap[NDSolve[
{
eq, nu, WhenEvent[Evaluate[conditions /. params], {Sow[V[t]],
If[Abs[r[t] - Rz] < 10 /. params, "StopIntegration", "CrossDiscontinuity"]}]]
} /. params, q, {t, 0, tk}
]
]
Out[ ]:= {{{V[t] → InterpolatingFunction[ Domain: {{0., 1040.}} Output: scalar][t],
r[t] → InterpolatingFunction[ Domain: {{0., 1040.}} Output: scalar][t],
θ[t] → InterpolatingFunction[ Domain: {{0., 1040.}} Output: scalar][t],
s[t] → InterpolatingFunction[ Domain: {{0., 1040.}} Output: scalar][t]}},
{{4990.156, 3004.331, 1235.302, 440.5153, 180.7772, 99.62761}}}
In[ ]:= velocities
Out[ ]:= {{4990.156, 3004.331, 1235.302, 440.5153, 180.7772, 99.62761}}
In[ ]:= tk = sol[[1, 1, 2, 0, 1, 1, 2]]
Out[ ]:= 1035.222
In[ ]:= Reap[Sum[Sow[i^2] + 1, {i, 10}]]
Out[ ]:= {395, {{1, 4, 9, 16, 25, 36, 49, 64, 81, 100}}}
```

## Траектория спуска

```
in[ ]:= ParametricPlot[{s[t], (r[t] - Rz) * 0.001 /. params /. sol, {t, 0, tk}},  
  FrameLabel -> {"s, км", "h, км"}, AspectRatio -> 1 / GoldenRatio]
```



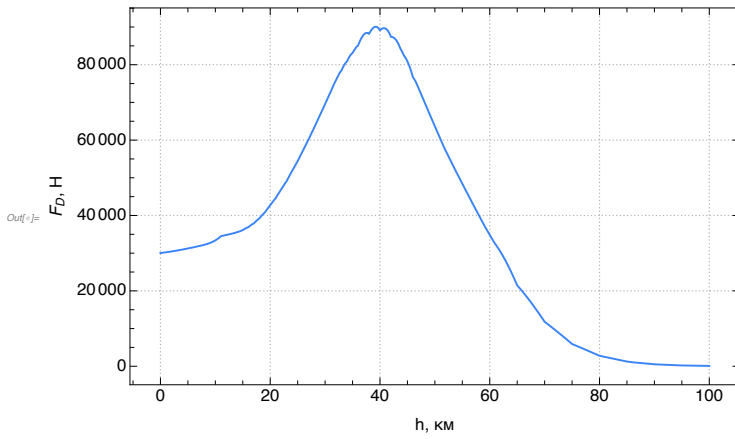
## Аэродинамическая сила

Аэродинамическая сила лобового сопротивления

```

In[ ]:= ParametricPlot[ { (r[t] - Rz) * 0.001, c_d S  $\frac{\rho[r[t] - Rz] V[t]^2}{2}$  } /. params /. sol, {t, 0, tk},
  AspectRatio -> 1/GoldenRatio, FrameLabel -> {"h, км", "F_D, Н"}, PlotRange -> All]

```



## Аэродинамическая сила

Максимум аэродинамической силы на траектории спускаемого аппарата

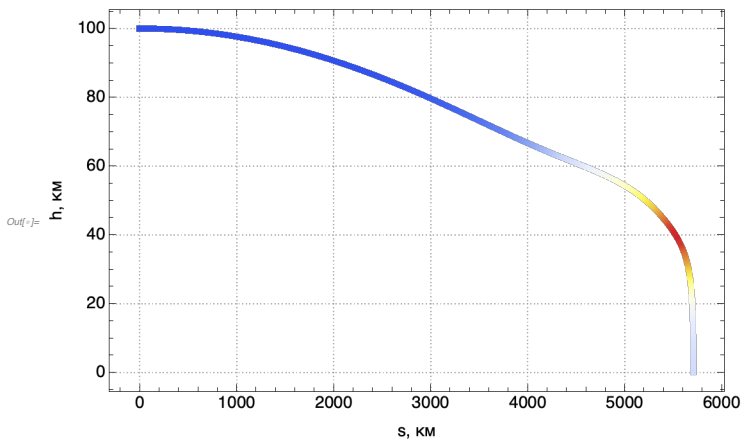
```
In[ ]:= FindMaximum[ $c_d S \frac{\rho[r[t] - R_z] V[t]^2}{2}$  /. params /. sol /. t -> tt, {tt, 100.0}]
```

```
Famax = %[[1]];
```

**FindMaximum:** The line search decreased the step size to within the tolerance specified by AccuracyGoal and PrecisionGoal but was unable to find a sufficient increase in the function. You may need more than MachinePrecision digits of working precision to meet these tolerances.

```
Out[ ]:= {90064.07, {tt -> 793.0997}}
```

```
In[ ]:= ParametricPlot[Flatten[{s[t], (r[t] - Rz)}] * 0.001 /. params /. sol,
  {t, 0, tk}, FrameLabel -> {"s, км", "h, км"}, AspectRatio -> 1/GoldenRatio,
  ColorFunction -> (ColorData["TemperatureMap"] [
     $c_d S ((\rho[r[t] - R_z] V[t]^2) / (2 F_{max}))$  /. params /. sol[[1]] /. t -> #3] &),
  ColorFunctionScaling -> False, PlotStyle -> Thickness[0.01]]
```



```
In[ ]:= ColorData["AvocadoColors"] [0.5]
```

```
Out[ ]:= ■
```

```
In[ ]:= ColorData["TemperatureMap"] [1]
```

```
Out[ ]:= ■
```

# Перегрузка

## Перегрузка

отношение абсолютной величины линейного ускорения, вызванного **негравитационными силами**, к ускорению свободного падения *на поверхности Земли*.

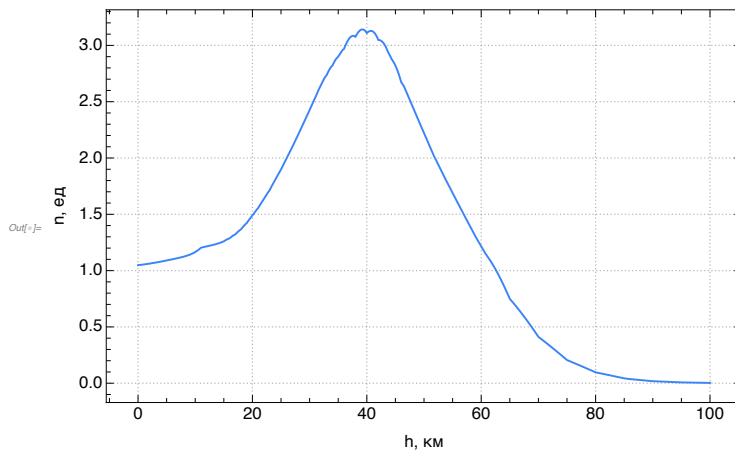
В рассматриваемой задаче негравитационные сила только одна -- это **аэродинамическая сила**

$$n = \frac{\sqrt{F_D^2 + F_L^2}}{m g}$$

$$n[t] := \frac{1}{g m} \text{Norm}\left[\{c_d, c_L\} S \frac{\rho[r[t] - Rz] V[t]^2}{2}\right];$$

```
ParametricPlot[{{(r[t] - Rz) * 0.001, %} /. params /. sol, {t, 0, tk},
```

```
AspectRatio -> 1/GoldenRatio, FrameLabel -> {"h, км", "n, ед"}, PlotRange -> All]
```



## Задание

1. Модифицируйте программу интегрирования уравнений движения спускаемого аппарата в атмосфере, добавив зависимость плотности воздуха от высоты, загружаемой из файла.
2. Модифицируйте программу интегрирования уравнений движения спускаемого аппарата в атмосфере, добавив зависимость ускорения свободного падения  $g$  от высоты. На сколько отличается дальность полета спускаемого аппарата при учете и без учета изменения ускорения свободного падения от высоты полета?
3. Постройте зависимость максимальной перегрузки, которую испытывает экипаж спускаемого аппарата в зависимости от начального угла входа в атмосферу ( $\theta_0$ ) в диапазоне от минус 15 до 0 градусов.
- 4. Постройте зависимость дальности точки приземления от начального угла входа в атмосферу ( $\theta_0$ ) в диапазоне от минус 15 до 0 градусов. Нарисуйте на одном графике траектории движения спускаемого аппарата при  $\theta_0 = -15^\circ$  и при  $\theta_0 = -2^\circ$ .**

## Пример

Функция расстояния до точки приземления в зависимости от начального угла наклона траектории.

```

In[ ]:= dist[θ0_?NumberQ] := Module[{params, nu, neq, tk, sol},
  params = {c_d → 1.3, c_L → 0.0, S →  $\frac{\pi 2.2^2}{4}$ , m → 3000.0,
    g → 9.807, Rz → 6371000.0, h0 → 100000.0, μ → 398600.4415 × 109};
  nu = {V[0] ==  $\sqrt{\mu / (Rz + h0)}$ , θ[0] == θ0, r[0] == Rz + h0, s[0] == 0};
  neq = eq /. params;
  sol = NDSolve[{eq, nu,
    WhenEvent[r[t] <= Rz /. params,
      "StopIntegration", DetectionMethod → "Sign"]} /. params,
    q, {t, 0, 5000}];
  tk = sol[[1, 1, 2, 0, 1, 1, 2]];
  s[t] /. sol /. t → tk
];
In[ ]:= dist[0.0°] * 0.001
... NDSolve: Event location failed to converge to the requested accuracy or precision within 100 iterations between t = 770.3860890330628 and t = 770.4553075372473.
Out[ ]:= {4543.227}

```

## Пример

```
In[ ]:= Plot[dist[ $\theta\theta^\circ$ ] * 0.001, { $\theta\theta$ , -15, 0}, PlotRange -> All, PlotPoints -> 2]
```

... **NDSolve**: Event location failed to converge to the requested accuracy or precision within 100 iterations between t = 169.0470994011732° and t = 169.09481465878406°.

Out[ ]:= \$Aborted

```
In[ ]:= Flatten[{# *  $\frac{180}{\pi}$ , dist[#] * 0.001}] & /@ Range[-15.0°, 0, 1°]
```

```
ListPlot[%, FrameLabel -> {" $\theta_0$ , ...°", "s, км"}, Joined -> True]
```

... **NDSolve**: Event location failed to converge to the requested accuracy or precision within 100 iterations between t = 168.96190399612527° and t = 169.00919261764543°.

... **NDSolve**: Event location failed to converge to the requested accuracy or precision within 100 iterations between t = 174.00913865484893° and t = 174.02645101327911°.

... **NDSolve**: Event location failed to converge to the requested accuracy or precision within 100 iterations between t = 179.32966096041383° and t = 179.56695057323404°.

... **General**: Further output of NDSolve::evcvmitt will be suppressed during this calculation.

```
Out[ ]:= {{-15., 310.7884}, {-14., 331.5668}, {-13., 355.1921}, {-12., 382.3201},
{-11., 413.825}, {-10., 450.8999}, {-9., 495.2154}, {-8., 549.187}, {-7., 616.4401},
{-6., 702.6765}, {-5., 817.4043}, {-4., 977.7401}, {-3., 1217.909}, {-2., 1617.774},
{-1., 2415.022}, {0., 4543.227}}
```

