Юдинцев В. В.

Самарский университет
04.03.2024

Лекция 3

Численные методыҷ

Решение линейных уравнений

In[ ]:=
```
ClearAll["Global`*"]
```

Уравнения в символьном виде

In[ ]:=
```
eq = {x + 2 y + z  == 1, 2 x - y - z == 2, z + y == 2};
Solve[eq, {x, y, z}]
```

Out[ ]=
```
{{x → 2, y → -3, z → 5}}
```

Решение линейных уравнений

Уравнения в матричной форме (задана матрица коэффициентов)

Решение уравнения вида
M x = B

In[ ]:=
```
ca = CoefficientArrays[eq, {x, y, z}] // Normal
```
Out[ ]=
```
{{-1, -2, -2}, {{1, 2, 1}, {2, -1, -1}, {0, 1, 1}}}
```

In[ ]:=
```
M = ca[[2]]; B = -ca[[1]];
LinearSolve[M, B]
```
Out[ ]=
```
{2, -3, 5}
```

Solve

*Функция Solve ищет **аналитическое** решение уравнения*

Например, все решения тригонометрического уравнения

In[ ]:=

```
Solve[Cos[x] + Sin[x]^2 == 0, x]
```

Out[ ]=

$$\left\{\left\{x \to \text{ConditionalExpression}\left[-\pi - \text{ArcTan}\left[\frac{\sqrt{2\left(-1+\sqrt{5}\right)}}{1-\sqrt{5}}\right] + 2\pi c_1,\ c_1 \in \mathbb{Z}\right]\right\},\ \left\{x \to \text{ConditionalExpression}\left[\pi + \text{ArcTan}\left[\frac{\sqrt{2\left(-1+\sqrt{5}\right)}}{1-\sqrt{5}}\right] + 2\pi c_1,\ c_1 \in \mathbb{Z}\right]\right\},\right.$$
$$\left.\left\{x \to \text{ConditionalExpression}\left[-i\,\text{ArcTanh}\left[\sqrt{\frac{2}{1+\sqrt{5}}}\right] + 2\pi c_1,\ c_1 \in \mathbb{Z}\right]\right\},\ \left\{x \to \text{ConditionalExpression}\left[i\,\text{ArcTanh}\left[\sqrt{\frac{2}{1+\sqrt{5}}}\right] + 2\pi c_1,\ c_1 \in \mathbb{Z}\right]\right\}\right\}$$

Solve

Только вещественные решения

In[ ]:=
```
sol = Solve[Cos[x] + Sin[x]^2 == 0, x, Reals]
```

Out[ ]=
$$\left\{\left\{x \to \text{ConditionalExpression}\left[-2\,\text{ArcTan}\left[\sqrt{2+\sqrt{5}}\,\right] + 2\,\pi\,c_1,\ c_1 \in \mathbb{Z}\right]\right\},\ \left\{x \to \text{ConditionalExpression}\left[2\,\text{ArcTan}\left[\sqrt{2+\sqrt{5}}\,\right] + 2\,\pi\,c_1,\ c_1 \in \mathbb{Z}\right]\right\}\right\}$$

ConditionalExpression означает, что решение представлено для c_1 принадлежащих целым числам ...-2, -1, 0, 1, 2, ...

Решение для c_1=0

In[ ]:=
```
sol /. c1 → 0
```

Out[ ]=
$$\left\{\left\{x \to -2\,\text{ArcTan}\left[\sqrt{2+\sqrt{5}}\,\right]\right\},\ \left\{x \to 2\,\text{ArcTan}\left[\sqrt{2+\sqrt{5}}\,\right]\right\}\right\}$$

NSolve

*Функция **NSolve** также ищет **аналитическое** решение уравнения и представляет результат в приближенном виде*

In[•]:=
```
NSolve[Cos[x] + Sin[x]^2 == 0, x]
```

Out[•]=
```
{{x → ConditionalExpression[1. (-2.23704 + 6.28319 C[1]), C[1] ∈ ℤ]}, {x → ConditionalExpression[1. (2.23704 + 6.28319 C[1]), C[1] ∈ ℤ]},
 {x → ConditionalExpression[1. ((0. - 1.06128 i) + 6.28319 C[1]), C[1] ∈ ℤ]}, {x → ConditionalExpression[1. ((0. + 1.06128 i) + 6.28319 C[1]), C[1] ∈ ℤ]}}
```

NSolve

Только вещественные решения

In[•]:=
```
sol = NSolve[Cos[x] + Sin[x]^2 == 0, x, Reals]
```

Out[•]=
```
{{x → ConditionalExpression[1. (-2.23704 + 6.28319 c1), c1 ∈ ℤ]}, {x → ConditionalExpression[1. (2.23704 + 6.28319 c1), c1 ∈ ℤ]}}
```

Решение для $c_1=0$

In[•]:=
```
sol /. c1 → 0
```

Out[•]=
```
{{x → -2.23704}, {x → 2.23704}}
```

NSolve

Функция NSolve пытается найти все решения уравнения.

In[ ]:=
```
NSolve[Cos[x] + Sin[x]^2 == 0, x]
```

Out[ ]=
```
{{x → ConditionalExpression[1. (-2.23704 + 6.28319 C[1]), C[1] ∈ ℤ]}, {x → ConditionalExpression[1. (2.23704 + 6.28319 C[1]), C[1] ∈ ℤ]},
 {x → ConditionalExpression[1. ((0. - 1.06128 i) + 6.28319 C[1]), C[1] ∈ ℤ]}, {x → ConditionalExpression[1. ((0. + 1.06128 i) + 6.28319 C[1]), C[1] ∈ ℤ]}}
```

К уравнениям могут быть добавлены дополнительные условия - неравенства, например:

In[ ]:=
```
NSolve[{Cos[x] + Sin[x]^2 == 0, x > -π, x < π}, x]
```

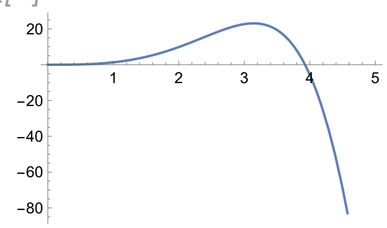
Out[ ]=
```
{{x → -2.23704}, {x → 2.23704}}
```

Solve/NSolve

Не все уравнения могут быть решены при помощи Solve (NSolve). Рассмотрим нелинейную функцию:

In[ ]:=
```
f = 2 Cosh[s] Sin[s] - 2 Cos[s] Sinh[s];
Plot[f, {s, 0, 5}]
```

Out[ ]=



Попытка решения нелинейного уравнения не приводит к результату.

In[ ]:=
```
NSolve[f, s]
```

NSolve: This system cannot be solved with the methods available to NSolve.

Out[ ]=
```
NSolve[2 Cosh[s] Sin[s] - 2 Cos[s] Sinh[s], s]
```

Решение нелинейных уравнений

Для решения нелинейных уравнений используется функция ***FindRoot***

Численное решение в окрестности начального приближения

Поиск решения в окрестности s = 2

In[•]:=
```
FindRoot[f, {s, 2.0}]
```
Out[•]=
$$\{s \to 2.20994 \times 10^{-8}\}$$

Поиск решения в окрестности s = 3.2

In[•]:=
```
FindRoot[f, {s, 3.2}]
```
Out[•]=
$$\{s \to 3.9266\}$$

Поиск корня в диапазоне от 2.5 до 5 и начальным приближением 3.5

In[•]:=
```
FindRoot[f, {s, 3.5, 3, 5}]
```
Out[•]=
$$\{s \to 3.9266\}$$

Решение нелинейных уравнений

Нет гарантии, что **FindRoot** найдет решение в заданном диапазоне (даже если оно там есть), все зависит от выбранного начального приближения

In[ ]:=
```
FindRoot[f, {s, 2.5, 2, 5}]
```

FindRoot: The point {2.} is at the edge of the search region {2., 5.} in coordinate 1 and the computed search direction points outside the region.

Out[ ]=
```
{s → 2.}
```

Функция при поиске корня вышла на границу интервала (решение не найдено), поскольку от начального приближения она двигалась в сторону убывания функции, тогда как решение находится “за холмом” - справа от начального приближения.

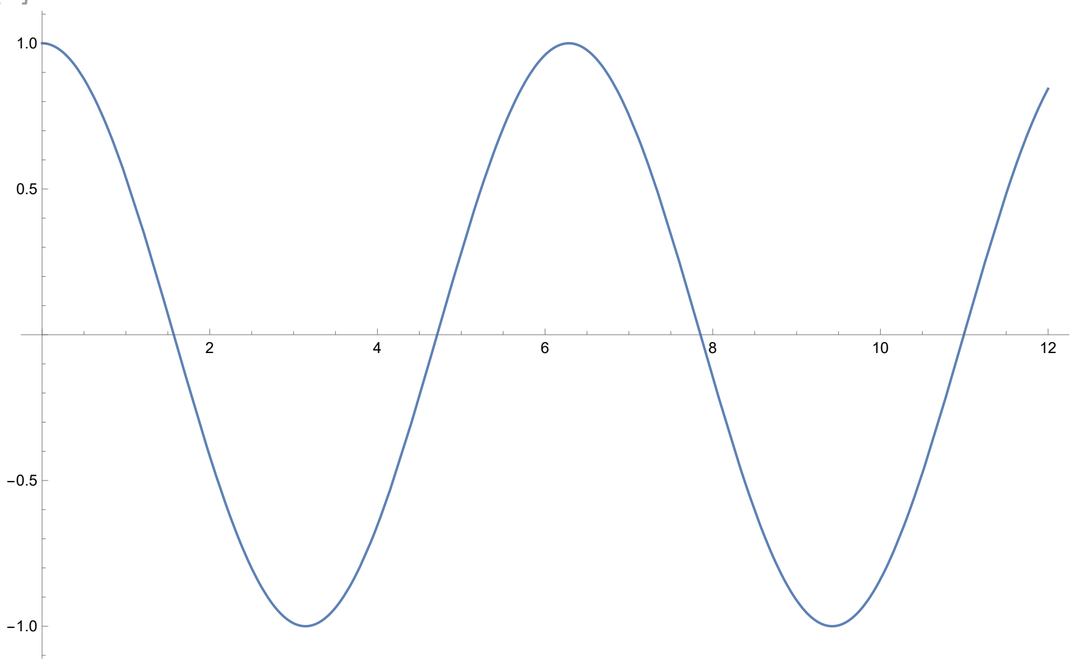
FindRoot

Решение не всегда ближайшее к начальному приближению

In[ ]:=
```
FindRoot[Cos[x] == 0, {x, 0.0001}]
```
Out[ ]=
```
{x → 10.9956}
```

In[ ]:=
```
FindRoot[Cos[x] == 0, {x, 1.0}]
```
Out[ ]=
```
{x → 1.5708}
```

In[ ]:=
```
Plot[Cos[x], {x, 0, 12}]
```
Out[ ]=



Пример

In[ ]:=

```
sx = NSolve[Cos[x] == Sin[x], x] /. c1 → # & /@ Range[-1, 1]
```

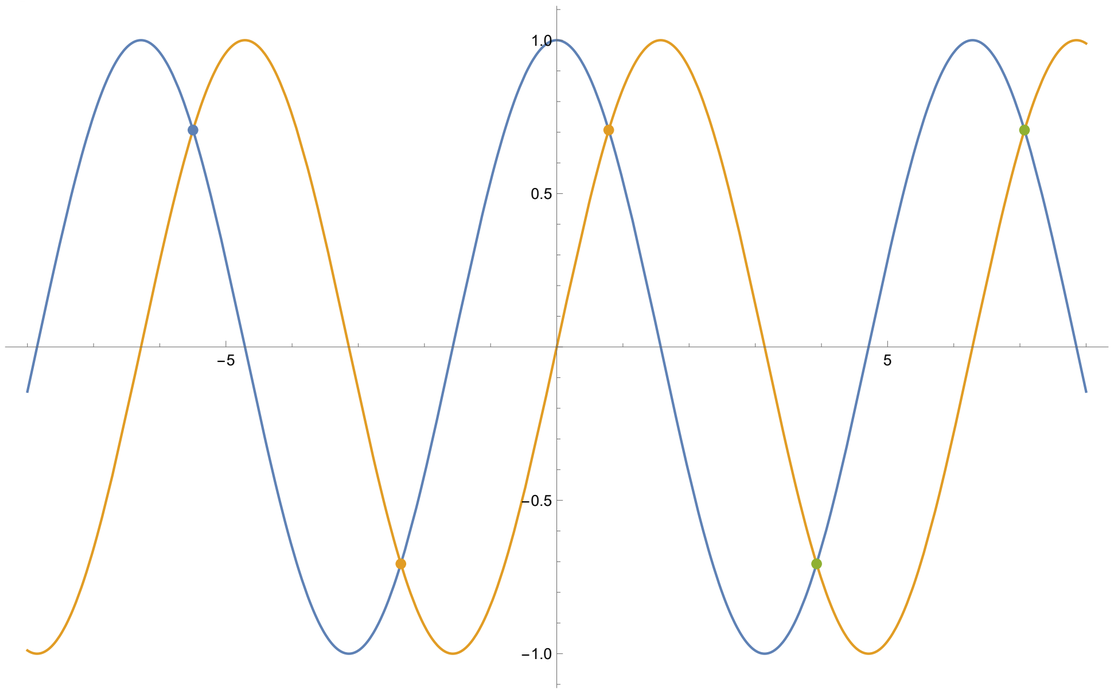
Out[ ]=

```
{{{x → -8.63938}, {x → -5.49779}}, {{x → -2.35619}, {x → 0.785398}}, {{x → 3.92699}, {x → 7.06858}}}
```

In[ ]:=

```
Show[
 Plot[{Cos[x], Sin[x]}, {x, -8, 8}],
 ListPlot[{x, Cos[x]} /. sx, PlotStyle → PointSize[Large]]
]
```

Out[ ]=



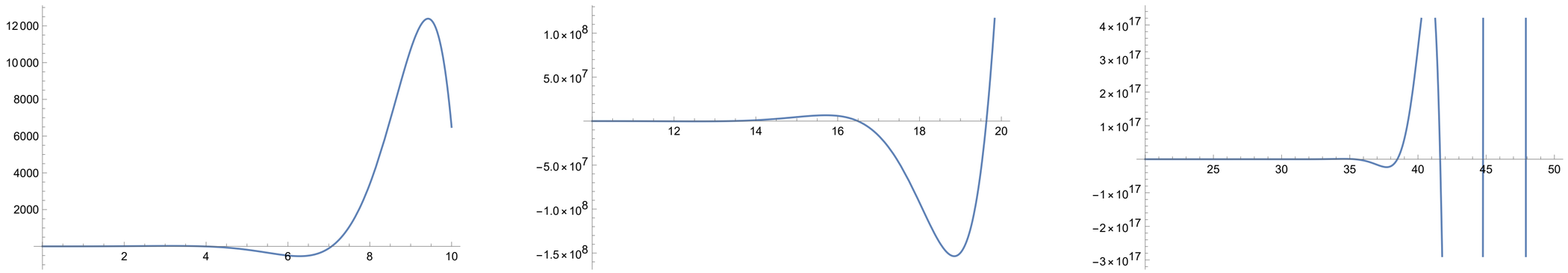
Пример поиска множества корней

Функция с бесконечным количеством корней

In[ ]:=

```
f = 2 Cosh[s] Sin[s] - 2 Cos[s] Sinh[s];
GraphicsRow[{Plot[f, {s, 0, 10}], Plot[f, {s, 10, 20}], Plot[f, {s, 20, 50}]}]
```

Out[ ]=



Пример поиска множества корней

Находим корни для массива начальных приближений в интервале от 0 до 20 с шагом 0,2. Получим список решений с повторениями.

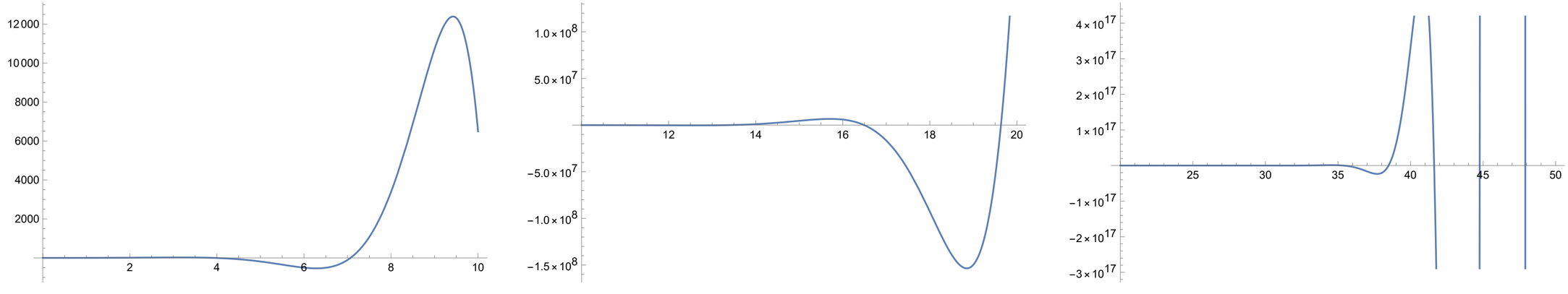
In[•]:=
```
FindRoot[f, {s, #}] & /@ Range[0, 20, 0.2]
```

Out[•]=

$$\{\{s \to 0.\}, \{s \to 2.1685 \times 10^{-8}\}, \{s \to 2.19683 \times 10^{-8}\}, \{s \to 9.66408 \times 10^{-9}\}, \{s \to 2.16778 \times 10^{-8}\}, \{s \to 1.79192 \times 10^{-8}\}, \{s \to 2.15327 \times 10^{-8}\},$$
$$\{s \to 5.70198 \times 10^{-9}\}, \{s \to 4.36725 \times 10^{-9}\}, \{s \to 1.54985 \times 10^{-9}\}, \{s \to 2.20994 \times 10^{-8}\}, \{s \to 1.2309 \times 10^{-8}\}, \{s \to 1.38749 \times 10^{-8}\}, \{s \to 2.39505 \times 10^{-8}\},$$
$$\{s \to 1.42215 \times 10^{-8}\}, \{s \to -5.39081 \times 10^{-9}\}, \{s \to 3.9266\}, \{s \to 3.9266\}, \{s \to 3.9266\}, \{s \to 3.9266\}, \{s \to 3.9266\}, \{s \to 3.9266\}, \{s \to 3.9266\},$$
$$\{s \to 3.9266\}, \{s \to 3.9266\}, \{s \to 3.9266\}, \{s \to 3.9266\}, \{s \to 3.9266\}, \{s \to 3.9266\}, \{s \to 3.9266\}, \{s \to 3.9266\}, \{s \to -1.79555 \times 10^{-8}\}, \{s \to 7.06858\},$$
$$\{s \to 7.06858\}, \{s \to 7.06858\}, \{s \to 7.06858\}, \{s \to 7.06858\}, \{s \to 7.06858\}, \{s \to 7.06858\}, \{s \to 7.06858\}, \{s \to 7.06858\}, \{s \to 7.06858\},$$
$$\{s \to 7.06858\}, \{s \to 7.06858\}, \{s \to 7.06858\}, \{s \to 7.06858\}, \{s \to 7.06858\}, \{s \to 7.06858\}, \{s \to 10.2102\}, \{s \to 10.2102\}, \{s \to 10.2102\},$$
$$\{s \to 10.2102\}, \{s \to 10.2102\}, \{s \to 10.2102\}, \{s \to 10.2102\}, \{s \to 10.2102\}, \{s \to 10.2102\}, \{s \to 10.2102\}, \{s \to 10.2102\}, \{s \to 10.2102\},$$
$$\{s \to 10.2102\}, \{s \to 10.2102\}, \{s \to 7.06858\}, \{s \to 13.3518\}, \{s \to 13.3518\}, \{s \to 13.3518\}, \{s \to 13.3518\}, \{s \to 13.3518\}, \{s \to 13.3518\},$$
$$\{s \to 13.3518\}, \{s \to 13.3518\}, \{s \to 13.3518\}, \{s \to 13.3518\}, \{s \to 13.3518\}, \{s \to 13.3518\}, \{s \to 13.3518\}, \{s \to 13.3518\}, \{s \to 13.3518\},$$
$$\{s \to 10.2102\}, \{s \to 16.4934\}, \{s \to 16.4934\}, \{s \to 16.4934\}, \{s \to 16.4934\}, \{s \to 16.4934\}, \{s \to 16.4934\}, \{s \to 16.4934\}, \{s \to 16.4934\},$$
$$\{s \to 16.4934\}, \{s \to 16.4934\}, \{s \to 16.4934\}, \{s \to 16.4934\}, \{s \to 16.4934\}, \{s \to 16.4934\}, \{s \to 16.4934\}, \{s \to 7.06858\}, \{s \to 19.635\}, \{s \to 19.635\},$$
$$\{s \to 19.635\}, \{s \to 19.635\}, \{s \to 19.635\}, \{s \to 19.635\}\}$$

Пример поиск множества корней

In[ ]:=
```
f = 2 Cosh[s] Sin[s] - 2 Cos[s] Sinh[s];
GraphicsRow[{Plot[f, {s, 0, 10}], Plot[f, {s, 10, 20}], Plot[f, {s, 20, 50}]}]
```
Out[ ]=



Если список решений рассмотреть как множество, то можно применить функцию Union, которая попытается исключить одинаковые элементы, однако поскольку корни вещественные числа, все дубли исключить не получится.

In[ ]:=
```
Union[{1, 2, 3, 3, 5, 5, 9}]
```
Out[ ]=
```
{1, 2, 3, 5, 9}
```

In[ ]:=
```
s /. FindRoot[f, {s, #}] & /@ Range[0, 20, 0.2] // Union
```
Out[ ]=
```
{-1.79555×10^-8, -5.39081×10^-9, 0., 1.54985×10^-9, 4.36725×10^-9, 5.70198×10^-9, 9.66408×10^-9, 1.2309×10^-8, 1.38749×10^-8, 1.42215×10^-8,
 1.79192×10^-8, 2.15327×10^-8, 2.16778×10^-8, 2.1685×10^-8, 2.19683×10^-8, 2.20994×10^-8, 2.39505×10^-8, 3.9266, 3.9266, 3.9266, 7.06858, 7.06858,
 7.06858, 7.06858, 10.2102, 10.2102, 10.2102, 10.2102, 13.3518, 13.3518, 13.3518, 16.4934, 19.635}
```

Пример поиск множества корней

Если допустить, что корни отличаются друг от друга не менее чем на 0,001, можно в начале округлить массив решений до 1 тысячной, а затем к округленным значениям применить функцию Union

In[•]:=
```
Round[s, 0.001] /. FindRoot[f, {s, #}] & /@ Range[0, 20, 0.2] // Union
```
Out[•]=
```
{0., 3.927, 7.069, 10.21, 13.352, 16.493, 19.635}
```

In[•]:=
```
roots = Round[s, 0.001] /. FindRoot[f, {s, #}] & /@ Range[0, 20, 0.2] // Union
```
Out[•]=
```
{0., 3.927, 7.069, 10.21, 13.352, 16.493, 19.635}
```

Таким образом в диапазоне от 0 до 20 у рассматриваемой функции 7 корней.

Линейное программирование

На заправочной станции два вида топлива (естественно, оно необычно):

1. «Антигравитон-10» (А-10) , 1 литр весит 3 кг
2. «Антигравитон-7» (А-7), 1 литр весит 2 кг

Цифра в названии топлива показывает, сколько часов можно пролететь на одном его литре.
Ограничение по массе топлива: не более 12 кг
Объем бака: 5 литров.

Какое же топливо выбрать? Очевидно, А-10 эффективнее, но 1 л его весит 3 кг, а общий вес топлива на борту нашего космолета не должен превышать 12 кг. Литр А-7 весит 2 кг. Можно залить им полный бак, он будет весить всего 10 кг, но запас хода окажется равным 35 часам.
Есть еще вариант: залить в бак сразу оба сорта топлива (они не перемешиваются) и использовать их поочередно. Но какого топлива сколько взять на борт?

Ограничение: **X1 + X2 <= 5**
Ограничение: **X1*3 + X2*2 <= 12**
Целевая функция: **максимум X1*10+X2*7**

Задача из журнал Юный техник, №8, 1985 год.

Линейное программирование

Ограничение: **X1 + X2 <= 5**
Ограничение: **X1*3 + X2*2 = 12**
Целевая функция: **максимум X1*10+X2*7**

Использование функции ***Maximize***

In[ ]:=
```
Maximize[{x1 * 10 + x2 * 7, x1 + x2 ≤ 5 && 3 x1 + 2 x2 == 12 && x1 > 0 && x2 > 0}, {x1, x2}]
```
Out[ ]=
```
{41, {x1 → 2, x2 → 3}}
```

Функция ***LinearProgramming***

Функция ищет минимум целевой функции при **m.x>b**

In[187]:=
```
solLP = LinearProgramming[-{10, 7}, -{{1, 1}, {3, 2}}, -{5, 12}] // N
```
Out[187]=
```
{2., 3.}
```

In[ ]:=
```
solLP.{10, 7}
```
Out[ ]=
```
41.
```

LinearProgramming[c, m, b]

finds a vector x that minimizes the quantity $c.x$ subject to the constraints $m.x \geq b$ and $x \geq 0$.

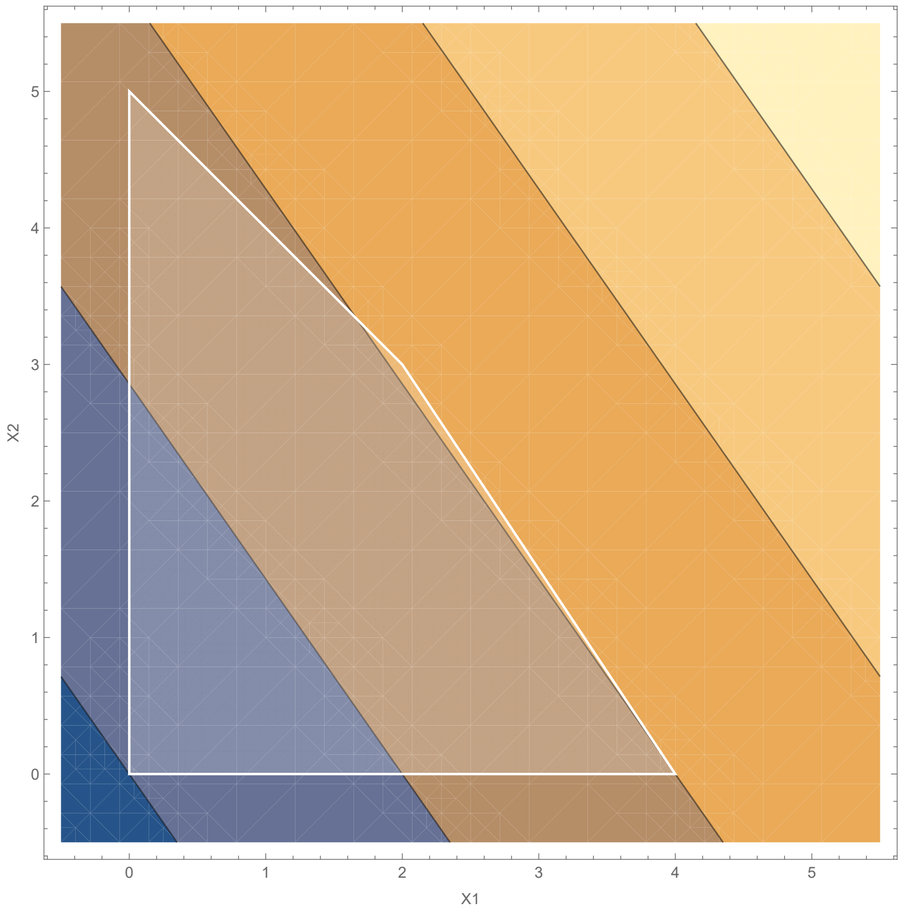
Линейное программирование

Геометрическое представление
Show, ContourPlot, RegionPlot

In[ ]:=

```
Show[
 ContourPlot[x1 * 10 + x2 * 7, {x1, -0.5, 5.5}, {x2, -0.5, 5.5}],
 RegionPlot[x1 + x2 ≤ 5 && x1 * 3 + x2 * 2 ≤ 12 && x1 > 0 && x2 > 0, {x1, -0.5, 5.5}, {x2, -0.5, 5.5}, PlotStyle → {White, Opacity[0.2]},
  BoundaryStyle → {White}, PlotPoints → 100],
 FrameLabel → {"X1", "X2"}
]
```

Out[ ]=

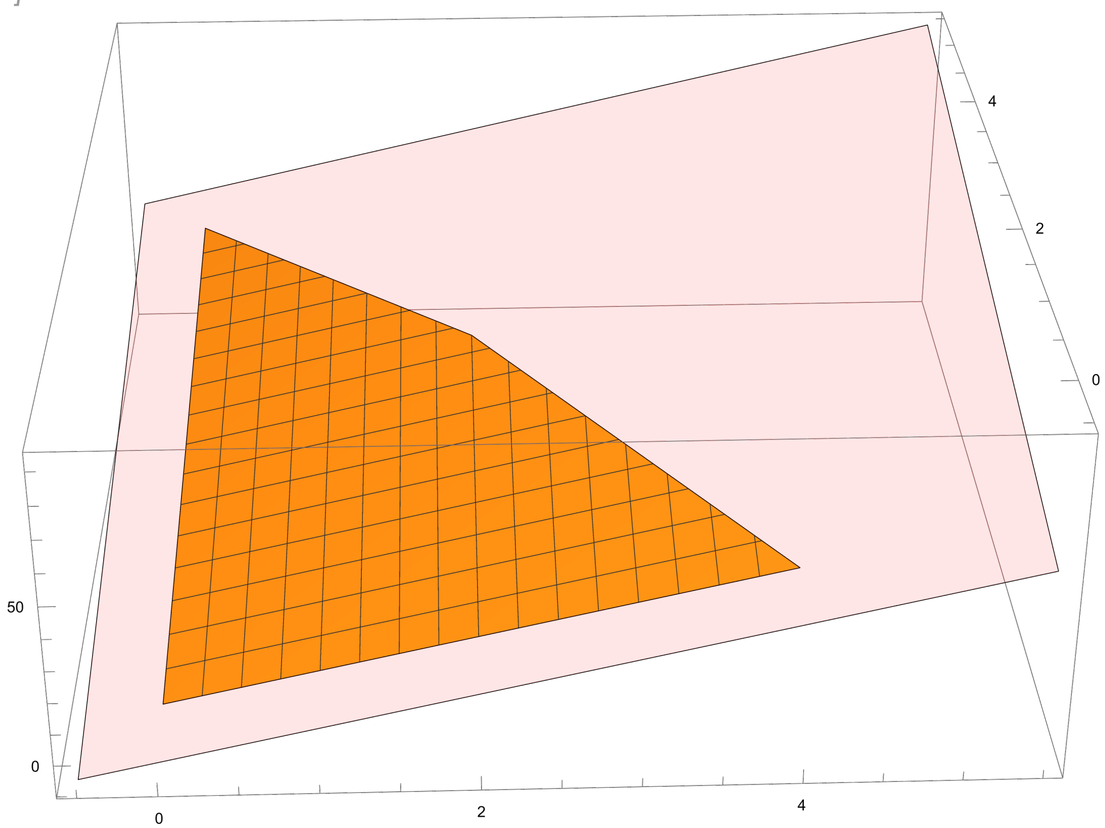


Линейное программирование

Геометрическое представление
Show, Plot3D

In[ ]:=
```
Show[
 Plot3D[x1 * 10 + x2 * 7, {x1, -0.5, 5.5}, {x2, -0.5, 5.5}, PlotStyle → Directive[Opacity[0.1], Red], Mesh → None],
 Plot3D[x1 * 10 + x2 * 7, {x1, -0.5, 5.5}, {x2, -0.5, 5.5}, RegionFunction → Function[{x1, x2}, x1 + x2 ≤ 5 && x1 * 3 + x2 * 2 ≤ 12 && x1 > 0 && x2 > 0],
  PlotPoints → 100]
]
```

Out[ ]=



Линейное программирование

Определить максимальный план производства при ограничениях:

Вид сырья	Продукт 1	Продукт 2	Продукт 3	Продукт 4	Запасы сырья
Сырьё 1	4 кг	2 кг	1 кг	8 кг	≤ 1200 кг
Сырьё 2	2 кг	10 кг	6 кг	0 кг	≤ 600 кг
Сырьё 3	3 кг	0 кг	6 кг	1 кг	≤ 1500 кг
Прибыль	15 р	6 р	12 р	24 р	максимум

In[ ]:=
```
Maximize[{15 x1 + 6 x2 + 12 x3 + 24 x4, 4 x1 + 2 x2 + 1 x3 + 8 x4 ≤ 1200 && 2 x1 + 10 x2 + 6 x3 + 0 x4 ≤ 600 && 3 x1 + 0 x2 + 6 x3 + 1 x4 ≤ 1500 && x1 > 0 && x2 > 0 &&
   x3 > 0 && x4 > 0}, {x1, x2, x3, x4}]
```
Out[ ]=
$$\left\{4500, \left\{x1 \to 0, x2 \to 0, x3 \to 100, x4 \to \frac{275}{2}\right\}\right\}$$

In[ ]:=
```
NMaximize[{15 x1 + 6 x2 + 12 x3 + 24 x4, 4 x1 + 2 x2 + 1 x3 + 8 x4 ≤ 1200 && 2 x1 + 10 x2 + 6 x3 + 0 x4 ≤ 600 && 3 x1 + 0 x2 + 6 x3 + 1 x4 ≤ 1500 && x1 > 0 && x2 > 0 &&
   x3 > 0 && x4 > 0}, {x1, x2, x3, x4}]
```
Out[ ]=
```
{4500., {x1 → 0., x2 → 0., x3 → 100., x4 → 137.5}}
```

Функция LinearProgramming

Функция ищет **минимум** целевой функции при **Mx>b**

In[ ]:=
```
LinearProgramming[-{15, 6, 12, 24}, -{{4, 2, 1, 8}, {2, 10, 6, 0}, {3, 0, 6, 1}}, -{1200, 600, 1500}] // N
```
Out[ ]=
```
{0., 0., 100., 137.5}
```

Максимум функции

FindMaximum

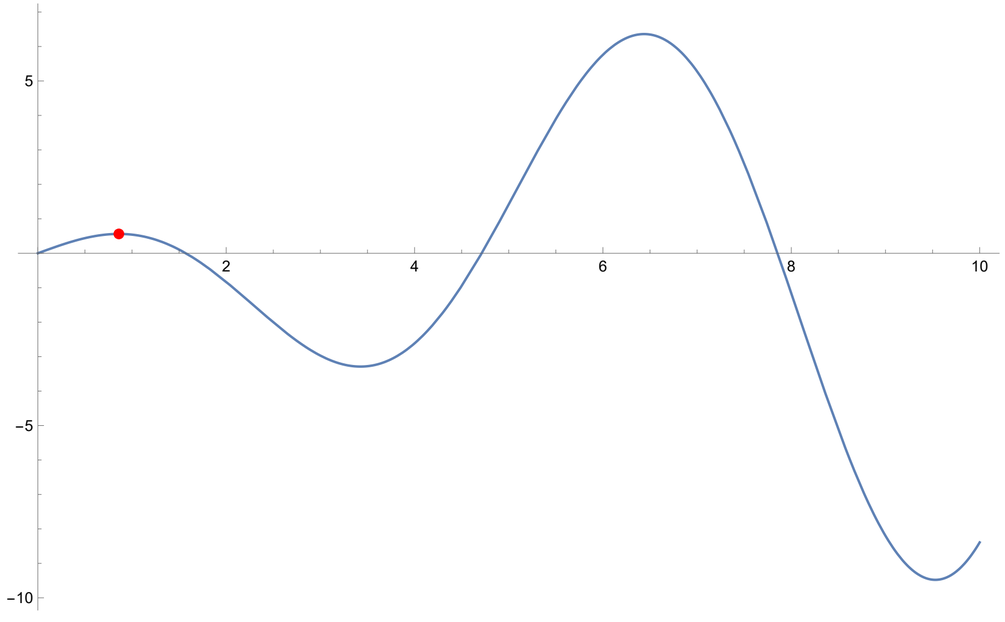
Функция **FindMaximum** ищет локальный максимум функции

In[ ]:=
```
f = x Cos[x];
```

Автоматический выбор начального приближения

In[ ]:=
```
max1 = FindMaximum[f, x]
```
Out[ ]=
```
{0.561096, {x → 0.860334}}
```

In[ ]:=
```
Plot[f, {x, 0, 10}, Epilog → {PointSize[Large], Red, Point[{x, f} /. max1[[2]]]}]
```
Out[ ]=

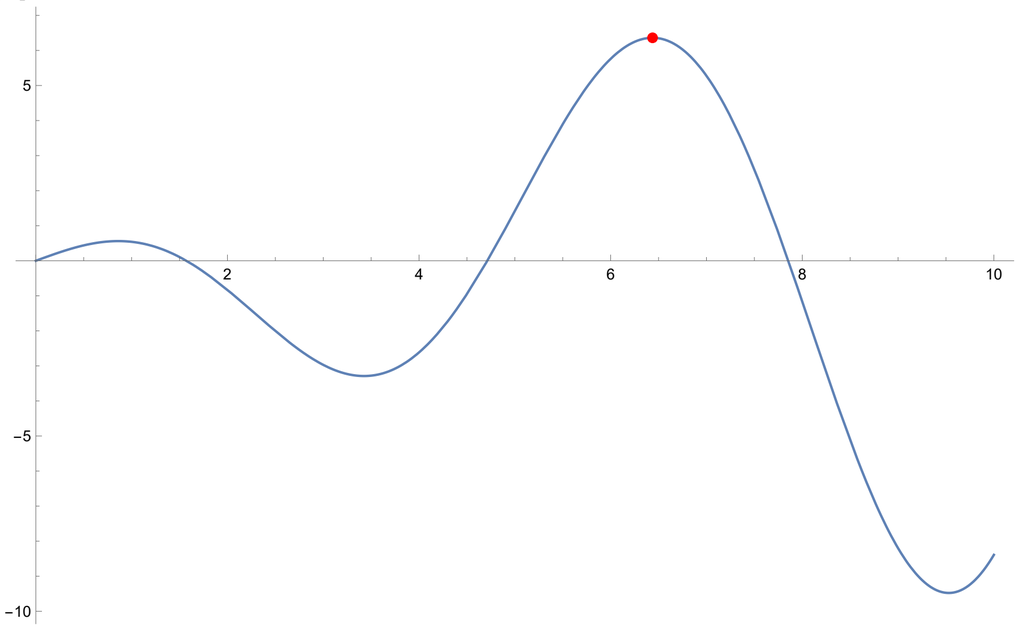


Максимум функции

Поищем в другом месте

In[ ]:=
```
max2 = FindMaximum[f, {x, 6}]
```
Out[ ]=
```
{6.361, {x → 6.4373}}
```

In[ ]:=
```
Plot[f, {x, 0, 10}, Epilog → {PointSize[Large], Red, Point[{x, f} /. max2[[2]]]}]
```
Out[ ]=



Дополнительно могут быть заданы ограничения, в этом случае максимум может быть достигнут на границе области

In[ ]:=
```
max2 = FindMaximum[{f, 0 < x < 6}, {x, 5}]
```
Out[ ]=
```
{5.76102, {x → 6.}}
```

Максимум функции

Maximize

Функция **Maximize** ищет глобальный максимум

Если функция неограниченная, то максимум не будет найден

In[ ]:=
```
max1 = Maximize[{f}, x]
```

Maximize: The maximum is not attained at any point satisfying the given constraints.

Out[ ]=
```
{∞, {x → -∞}}
```

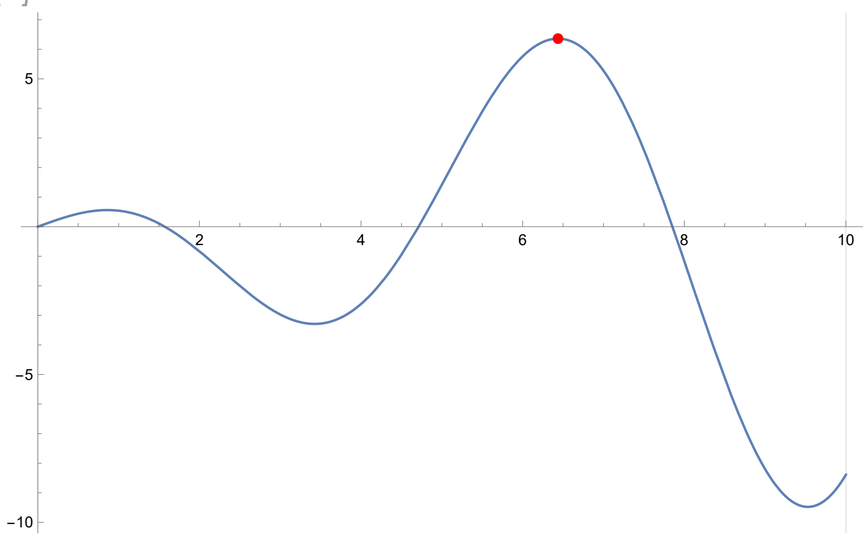
Необходимы дополнительные ограничения

In[ ]:=
```
max1 = Maximize[{f, 0 < x < 10}, x] // N
```

Out[ ]=
```
{6.361, {x → 6.4373}}
```

In[ ]:=
```
Plot[f, {x, 0, 10}, Epilog → {PointSize[Large], Red, Point[{x, f} /. max1[[2]]]}, GridLines → {{0, 10}, None}]
```

Out[ ]=



Максимум функции

Результат может лежать на границе

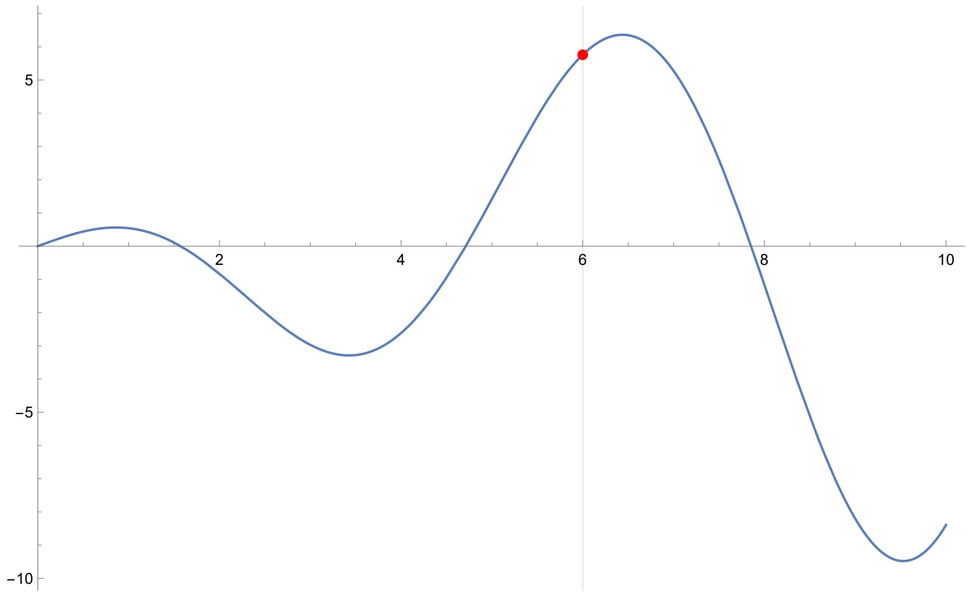
In[ ]:=
```
max1 = Maximize[{f, 0 < x < 6}, x] // N
```

Maximize: Warning: there is no maximum in the region in which the objective function is defined and the constraints are satisfied; a result on the boundary will be returned.

Out[ ]=
```
{5.76102, {x → 6.}}
```

In[ ]:=
```
Plot[f, {x, 0, 10}, Epilog → {PointSize[Large], Red, Point[{x, f} /. max1[[2]]]}, GridLines → {{0, 6}, None}]
```

Out[ ]=



Интерполяция

Линейная интерполяция

In[ ]:=
```
f =.;
data = {{0, 1}, {1, 2}, {2, 4}};
f[x_] = Interpolation[data, x, InterpolationOrder → 1]
f[0.5]
```

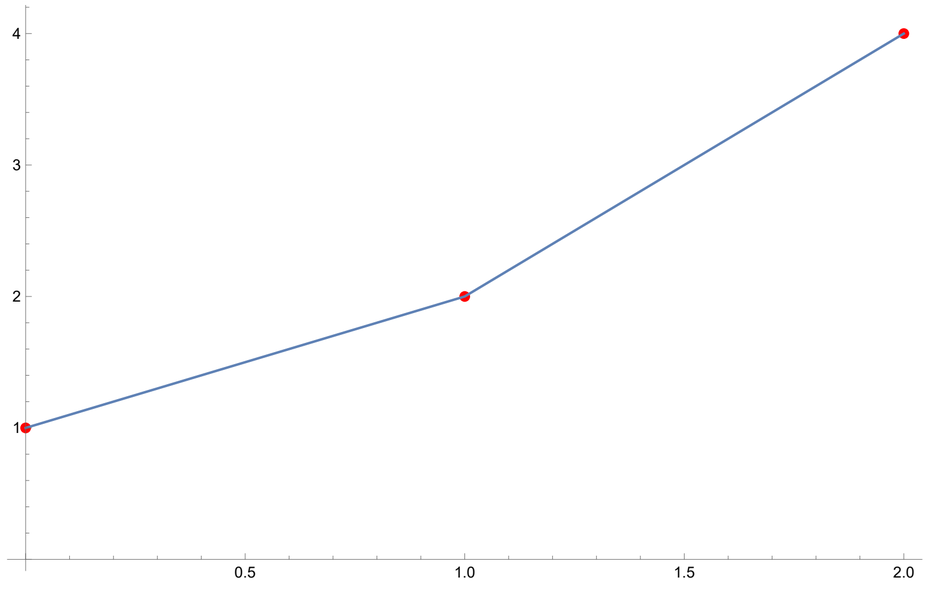
Out[ ]=

InterpolatingFunction [Domain: {{0, 2}} Output: scalar] [x]

Out[ ]=
```
1.5
```

In[ ]:=
```
Show[
 ListPlot[data, PlotStyle → {Red, PointSize[Large]}],
 Plot[f[x], {x, 0, Max[Transpose[data][[1]]]}]
]
```

Out[ ]=



Интерполяция

По умолчанию используется сплайн-интерполяция -- кусочная интерполяция полиномами 3 степени (если количество точек данных достаточно для этого)

In[ ]:=
```
f =.;
data = {{0, 1}, {1, 2}, {2, 4}, {3, 6}, {4, 6.7}};
f[x_] = Interpolation[data, x]
f[0.5]
```

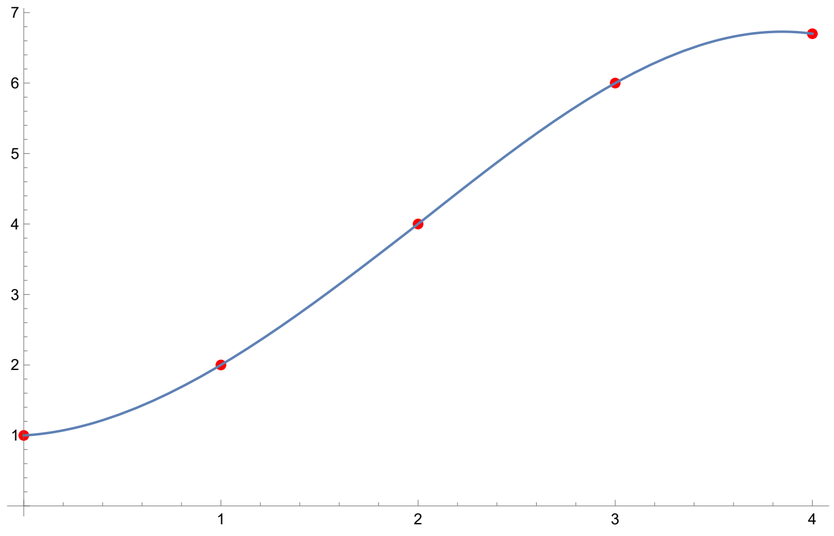
Out[ ]=

InterpolatingFunction [Domain: {{0., 4.}} Output: scalar] [x]

Out[ ]=
```
1.3125
```

In[ ]:=
```
Show[
 ListPlot[data, PlotStyle → {Red, PointSize[Large]}],
 Plot[f[x], {x, 0, Max[Transpose[data][[1]]]}],
 PlotLegends → Placed[{"dsd", "1"}, {Right, Top}]
]
```

Out[ ]=



Интерполяция данных из таблицы-файла

Импорт таблицы из текстового файла. Первый две строки файла содержат описание столбцов таблицы.

In[•]:=
```
NotebookDirectory[]
```
Out[•]=
```
/Users/vadim/Documents/Classes/Интегрированные_мат_пакеты/mathematica/
```

In[•]:=
```
data = Import[NotebookDirectory[] <> "/atm.txt", "Table"];
data[[1 ;; 10]] // TableForm
```
Out[•]//TableForm=

alt	sigma	delta	theta	temp	press	dens	a	visc	k.visc
km	K	N/sq.m	kg/cu.m	m/s	kg/m-s	sq.m/s			
-2	1.2067	1.2611	1.0451	301.2	127 800.	1.478	347.9	18.51	0.0000125
0	1.	1.	1.	288.1	101 300.	1.225	340.3	17.89	0.0000146
2	0.82168	0.78462	0.9549	275.2	79 500.	1.007	332.5	17.26	0.0000171
4	0.66885	0.60854	0.9098	262.2	61 660.	0.8193	324.6	16.61	0.0000203
6	0.53887	0.466	0.8648	249.2	47 220.	0.6601	316.5	15.95	0.0000242
8	0.42921	0.35185	0.8198	236.2	35 650.	0.5258	308.1	15.27	0.000029
10	0.33756	0.26153	0.7748	223.3	26 500.	0.4135	299.5	14.58	0.0000353
12	0.25464	0.19146	0.7519	216.6	19 400.	0.3119	295.1	14.22	0.0000456

Зависимость температуры от высоты

In[ ]:=
```
data[[3 ;;, {1, 5}]]
temperature[h_] = Interpolation[%][h]
```

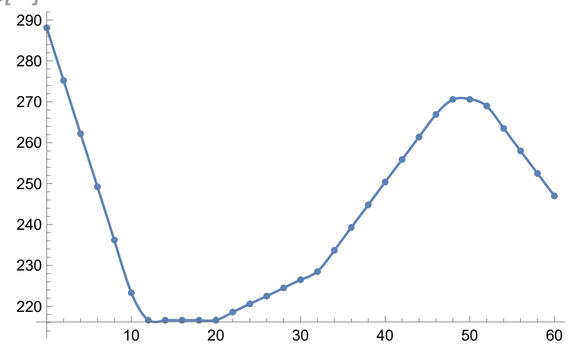
Out[ ]=
```
{{-2, 301.2}, {0, 288.1}, {2, 275.2}, {4, 262.2}, {6, 249.2}, {8, 236.2}, {10, 223.3}, {12, 216.6}, {14, 216.6}, {16, 216.6}, {18, 216.6},
 {20, 216.6}, {22, 218.6}, {24, 220.6}, {26, 222.5}, {28, 224.5}, {30, 226.5}, {32, 228.5}, {34, 233.7}, {36, 239.3}, {38, 244.8}, {40, 250.4},
 {42, 255.9}, {44, 261.4}, {46, 266.9}, {48, 270.6}, {50, 270.6}, {52, 269.}, {54, 263.5}, {56, 258.}, {58, 252.5}, {60, 247.}, {62, 241.5},
 {64, 236.}, {66, 230.5}, {68, 225.1}, {70, 219.6}, {72, 214.3}, {74, 210.3}, {76, 206.4}, {78, 202.5}, {80, 198.6}, {82, 194.7}, {84, 190.8},
 {86, 186.9}}
```

Out[ ]=

InterpolatingFunction [ Domain: {{-2., 86.}} Output: scalar] [h]

In[ ]:=
```
Show[
 Plot[temperature[h], {h, 0, 60}],
 ListPlot[data[[3 ;;, {1, 5}]]]
]
```

Out[ ]=



In[ ]:=

Вектор-функция

Импорт табличных данных из файла. Функция **Import**

In[ ]:=
```
data = Import[NotebookDirectory[] <> "/atm.txt", "Table"];
data[[1 ;; 5]] // TableForm
```

Out[ ]//TableForm=

alt	sigma	delta	theta	temp	press	dens	a	visc	k.visc
km	K	N/sq.m	kg/cu.m	m/s	kg/m-s	sq.m/s			
-2	1.2067	1.2611	1.0451	301.2	127 800.	1.478	347.9	18.51	0.0000125
0	1.	1.	1.	288.1	101 300.	1.225	340.3	17.89	0.0000146
2	0.82168	0.78462	0.9549	275.2	79 500.	1.007	332.5	17.26	0.0000171

Заголовок таблицы занимает первые две строки файла, игнорируем их **{3;;}**.
Выбираем столбцы 1, 5 и 8. Превращаем список троек в список значений пар: значение аргумента, значения вектор-функции

In[ ]:=
```
{#[[1]], {#[[2]], #[[3]]}} & /@ data[[3 ;;, {1, 5, 8}]]

tempSpeed[h_] = Interpolation[%][h]
```

Out[ ]=
```
{{-2, {301.2, 347.9}}, {0, {288.1, 340.3}}, {2, {275.2, 332.5}}, {4, {262.2, 324.6}}, {6, {249.2, 316.5}}, {8, {236.2, 308.1}},
 {10, {223.3, 299.5}}, {12, {216.6, 295.1}}, {14, {216.6, 295.1}}, {16, {216.6, 295.1}}, {18, {216.6, 295.1}}, {20, {216.6, 295.1}},
 {22, {218.6, 296.4}}, {24, {220.6, 297.7}}, {26, {222.5, 299.1}}, {28, {224.5, 300.4}}, {30, {226.5, 301.7}}, {32, {228.5, 303.}},
 {34, {233.7, 306.5}}, {36, {239.3, 310.1}}, {38, {244.8, 313.7}}, {40, {250.4, 317.2}}, {42, {255.9, 320.7}}, {44, {261.4, 324.1}},
 {46, {266.9, 327.5}}, {48, {270.6, 329.8}}, {50, {270.6, 329.8}}, {52, {269., 328.8}}, {54, {263.5, 325.4}}, {56, {258., 322.}},
 {58, {252.5, 318.6}}, {60, {247., 315.1}}, {62, {241.5, 311.5}}, {64, {236., 308.}}, {66, {230.5, 304.4}}, {68, {225.1, 300.7}},
 {70, {219.6, 297.1}}, {72, {214.3, 293.4}}, {74, {210.3, 290.7}}, {76, {206.4, 288.}}, {78, {202.5, 285.3}}, {80, {198.6, 282.5}},
 {82, {194.7, 279.7}}, {84, {190.8, 276.9}}, {86, {186.9, 274.1}}}
```

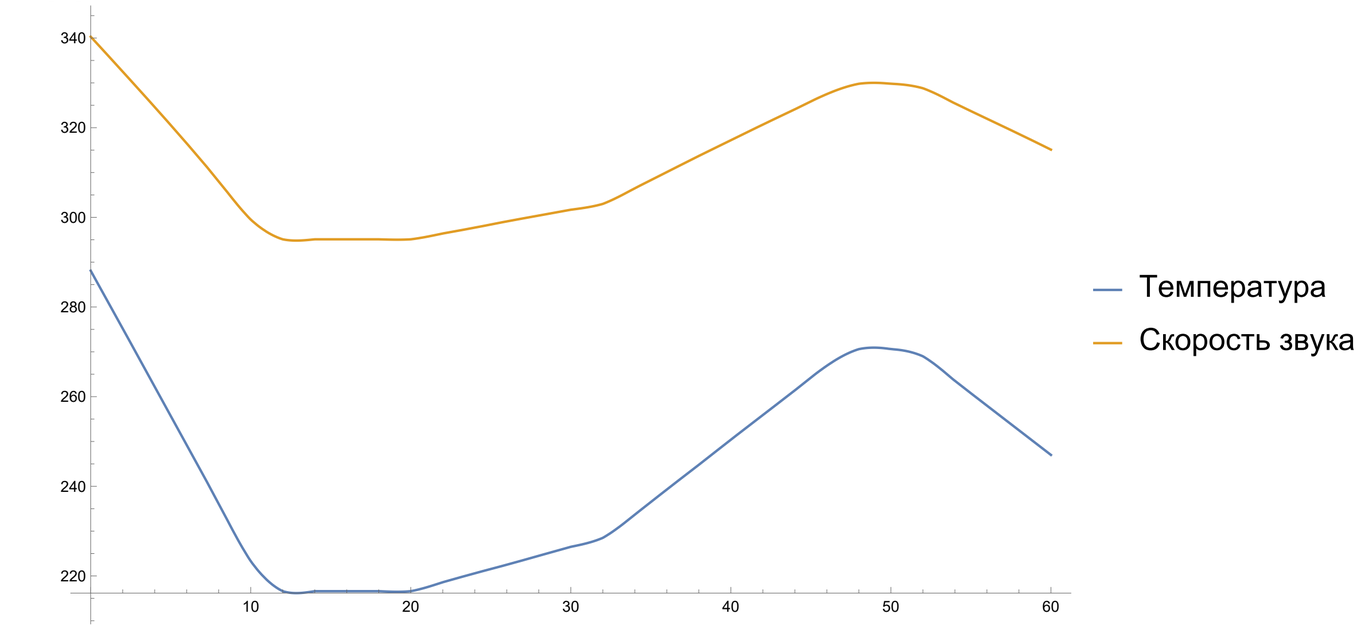
In[ ]:=
```
tempSpeed[35]
```

Out[ ]=
```
{236.481, 308.294}
```

Вектор-функция

In[ ]:=
```
tempSpeed[10]
```
Out[ ]=
```
{223.3, 299.5}
```

In[ ]:=
```
Plot[{tempSpeed[h][[1]], tempSpeed[h][[2]]}, {h, 0, 60}, PlotLegends → {"Температура", "Скорость звука"}]
```
Out[ ]=



In[ ]:=