

ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

КАФЕДРА АВТОМАТИЗАЦИЯ ПРОИЗВОДСТВЕННЫХ ПРОЦЕССОВ

ЛЕКЦИЯ №11

Математическая библиотека

scipy

и библиотека визуализации данных

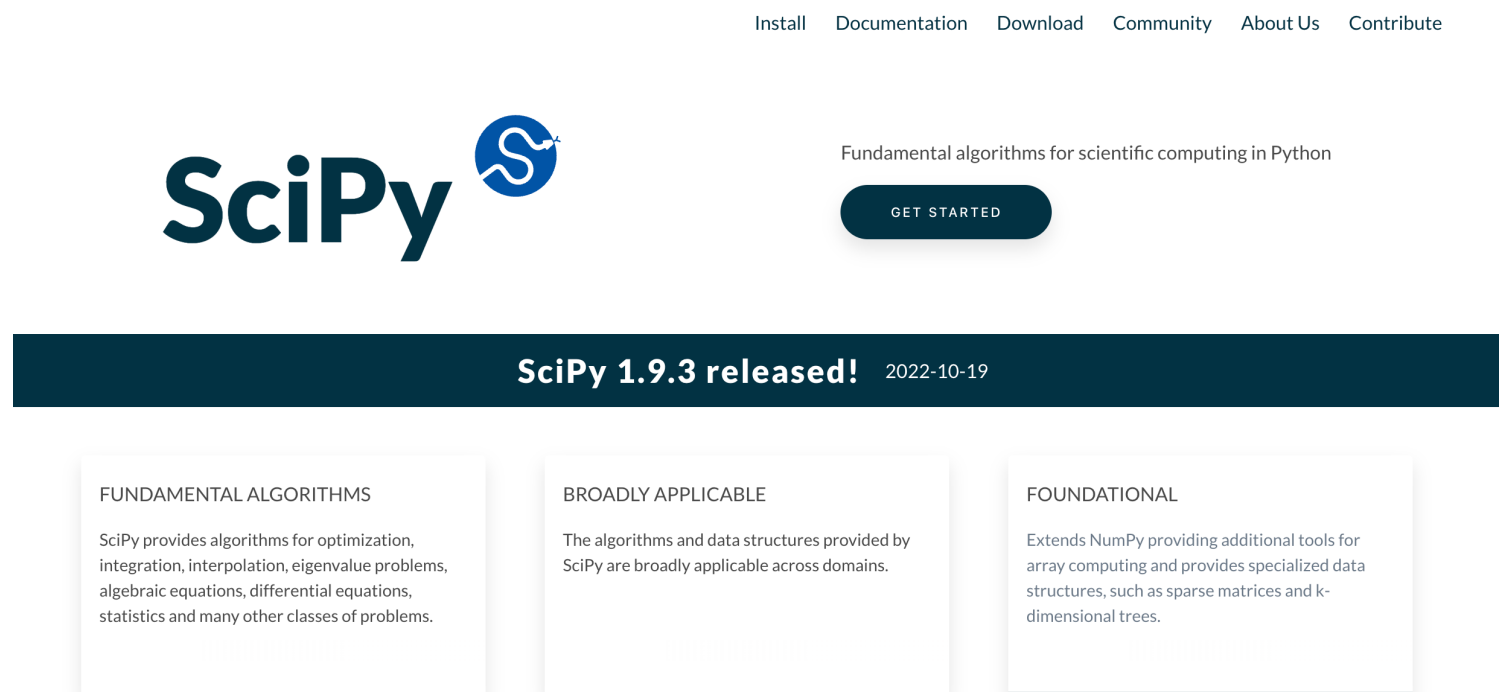
matplotlib

СОСТАВИТЕЛЬ: КАНД. ТЕХН. НАУК БЫКАДОР В.С.

Общее описание математической библиотеки **scipy**

Математическая библиотека **scipy** является широко используемой библиотекой для выполнения численных расчётов с использованием языка программирования **Python**. В данной библиотеке содержится обширное количество функций для выполнения численного интегрирования, дифференцирования, решения задач оптимизации и интерполяции, прямого и обратного преобразований Фурье и ряд других.

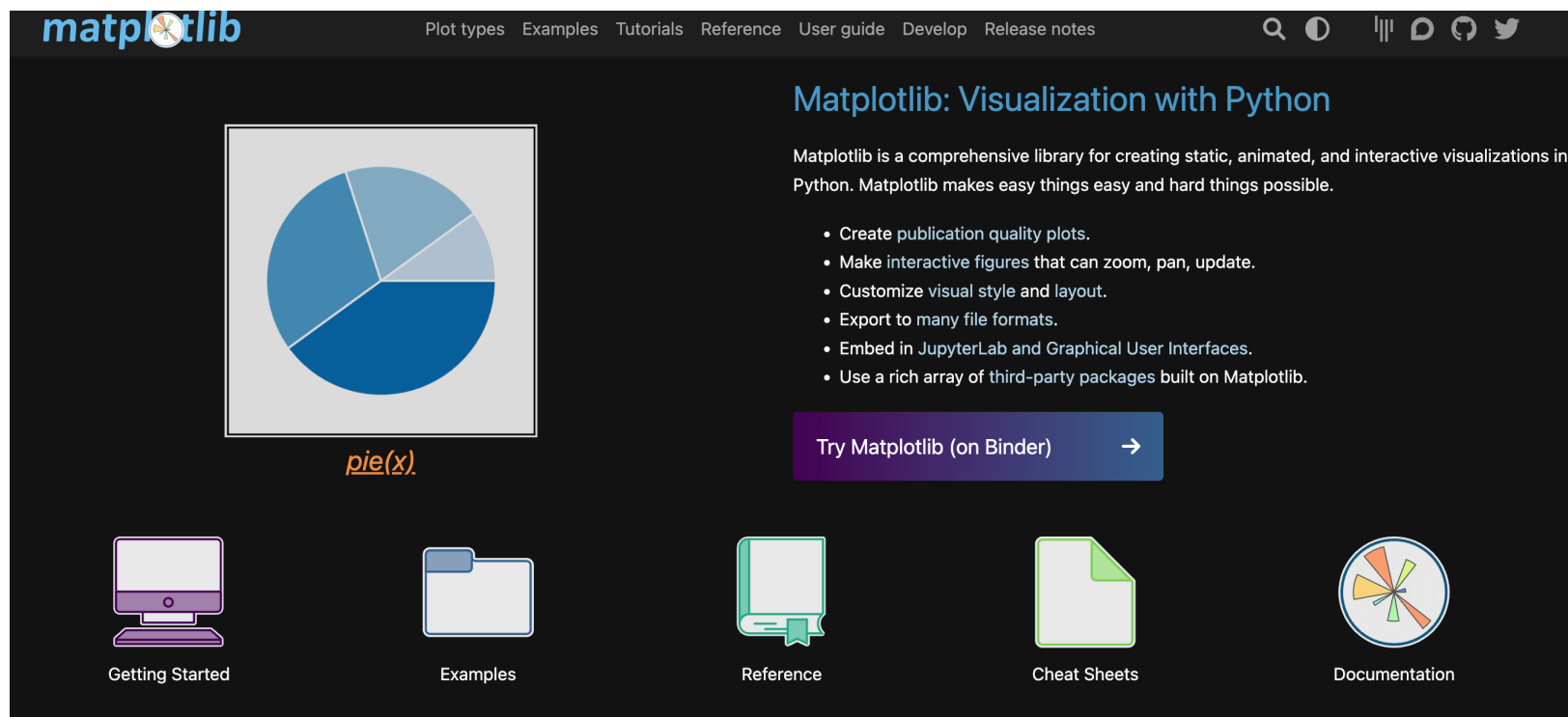
<https://scipy.org/>



Общее описание библиотеки `matplotlib` визуализации данных

Библиотека визуализации данных `matplotlib` широко для визуального отображения различных численных данных в виде графиков. Данная библиотека имеет различные виды графиков, а также позволяет настраивать различные параметры графиков (толщина линии, цвет, вид маркеров и типов линий и др.).

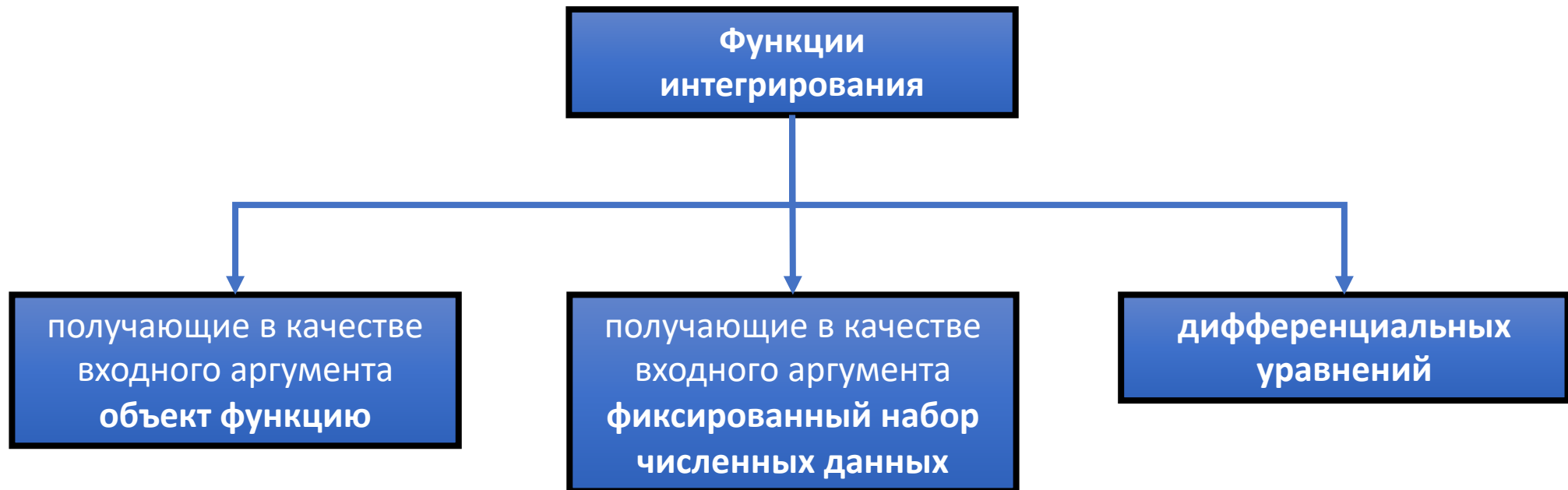
<https://matplotlib.org/>



Basic plots	
	<code>plot([X], Y, [fmt], ...)</code> X, Y, fmt, color, marker, linestyle API
	<code>scatter(X, Y, ...)</code> X, Y, [s]izes, [c]olors, marker, cmap API
	<code>bar(h)(x, height, ...)</code> x, height, width, bottom, align, color API
	<code>imshow(Z, ...)</code> Z, cmap, interpolation, extent, origin API
	<code>contour[f]([X], [Y], Z, ...)</code> X, Y, Z, levels, colors, extent, origin API
	<code>pcolormesh([X], [Y], Z, ...)</code> X, Y, Z, vmin, vmax, cmap API
	<code>quiver([X], [Y], U, V, ...)</code> X, Y, U, V, C, units, angles API
	<code>pie(X, ...)</code> Z, explode, labels, colors, radius API
	<code>text(x, y, text, ...)</code> x, y, text, va, ha, size, weight, transform API
	<code>fill[_between]([x] (...))</code> X, Y1, Y2, color, where API
Advanced plots	
	<code>step(X, Y, [fmt], ...)</code> X, Y, fmt, color, marker, where API
	<code>boxplot(X, ...)</code> X, notch, sym, bootstrap, widths API
	<code>errorbar(X, Y, xerr, yerr, ...)</code> X, Y, xerr, yerr, fmt API
	<code>hist(X, bins, ...)</code> X, bins, range, density, weights API
	<code>violinplot(D, ...)</code> D, positions, widths, vert API
	<code>barbs([X], [Y], U, V, ...)</code> X, Y, U, V, C, length, pivot, sizes API
	<code>eventplot(positions, ...)</code> positions, orientation, lineoffsets API
	<code>hexbin(X, Y, C, ...)</code> X, Y, C, gridsize, bins API

Численное интегрирование

Существует несколько функций интегрирования в библиотеке `scipy.integrate`. Эти функции интегрирования можно разделить на три категории.



Интегрирование функций

$$I = \int_0^2 x^2 dx = \frac{x^3}{3} \Big|_0^2 = \frac{1}{3}(2^3 - 0^3) = \frac{8}{3} \approx 2,67$$

$$I = \int_0^1 e^{-x} dx = \frac{1}{-1} \int_0^1 e^{-x} dx = -e^{-x} \Big|_0^1 = -(e^{-1} - e^{-0}) \approx 0,63$$

```
1 from scipy.integrate import quad
2
3 def f(x):
4     return x**2
5
6 I = quad(f, 0, 2)
7
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ

```
• (venv) vityalybykador@Air-Vitaly test % /Users/vitaly
or/PROJECTS/test/for_lectures2.py

(2.6666666666666665, 2.9605947323337504e-14)
```

Результат интегрирования

Оценка ошибки интегрирования

```
1 import numpy as np
2 from scipy.integrate import quad
3
4 def f(x):
5     return np.exp(-x)
6
7 I = quad(f, 0, 1)
8
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ

```
• (venv) vityalybykador@Air-Vitaly test % /Users/vitaly
or/PROJECTS/test/for_lectures2.py

(0.6321205588285577, 7.017947987503855e-15)
```

Интегрирование функций с параметром

$$I = \int_0^5 (a \cdot x) dx = a \cdot \frac{x^2}{2} \Big|_0^5 = a \cdot 12,5$$

$$a = 2 \Rightarrow 25$$

```
1 from scipy.integrate import quad
2
3 def f(x, a):
4     return a*x
5
6 I = quad(f, 0, 5, args=(2,))
7
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ

• (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for_lectures2.py

(25.0, 2.7755575615628914e-13)

$$a = 3 \Rightarrow 37,5$$

```
1 from scipy.integrate import quad
2
3 def f(x, a):
4     return a*x
5
6 I = quad(f, 0, 5, args=(3,))
7
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ

• (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for_lectures2.py

(37.49999999999999, 4.163336342344336e-13)

Интегрирование функций с параметром

$$I = \int_0^2 (a \cdot x^b) dx$$

```
1  from scipy.integrate import quad
2
3  def f(x, a, b):
4      return a*(x**b)
5
6  I = quad(f, 0, 2, args=(2, 4,))
7
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

```
• (venv) vitallybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for_lectures2.py
```

```
(12.8, 1.4210854715202004e-13)
```

Интегрирование функций с параметром (двойной интеграл)

$$D = \int_3^5 \int_0^2 (a \cdot x \cdot y) dy dx$$

$a = 10$

```
1  from scipy.integrate import dblquad
2
3  def f(y, x, a):
4      return a*x*y
5
6  I = dblquad(f, 3, 5, 0, 2, args=(10,))
7
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

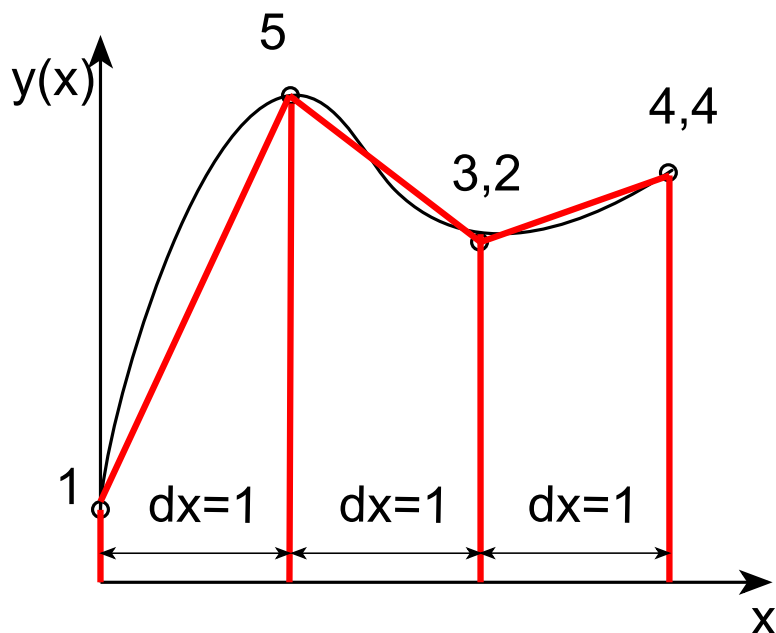
КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

```
• (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for_lectures2.py
```

```
(160.0, 1.7763568394002505e-12)
```


Интегрирование фиксированного набора данных



```
1  from scipy.integrate import trapezoid
2
3  y = [1, 5, 3.2, 4.4]
4
5
6  I = trapezoid(y)
7
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

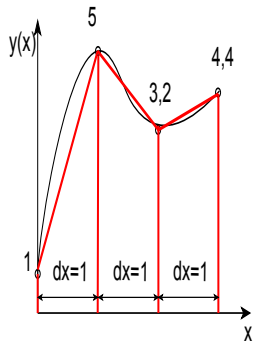
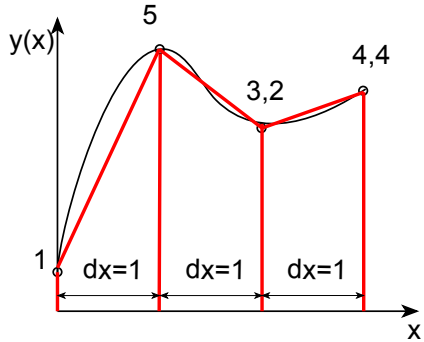
КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

```
● (venv) vitalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for_lectures2.py
```

10.9

Интегрирование фиксированного набора данных



```
1 from scipy.integrate import trapezoid
2
3 y = [1, 5, 3.2, 4.4]
4
5
6 I = trapezoid(y, dx=1)
7
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ

• (venv) vityalybykador@Air-Vitaly test % or/PROJECTS/test/for_lectures2.py

10.9

```
1 from scipy.integrate import trapezoid
2
3 y = [1, 5, 3.2, 4.4]
4
5
6 I = trapezoid(y, dx=0.5)
7
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ

• (venv) vityalybykador@Air-Vitaly test % /Users/vityalybykador/PROJECTS/test/for_lectures2.py

5.45

Интегрирование фиксированного набора данных

```
1  from scipy.integrate import trapezoid
2
3  y = [1, 5, 3.2, 4.4]
4  x = [3, 5, 7, 9]
5
6
7  I = trapezoid(y, x=x)
8
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ

```
• (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for_lectures2.py
```

21.8

```
1  from scipy.integrate import trapezoid
2
3  y = [1, 5, 3.2, 4.4]
4  x = [3, 5, 7, 9]
5
6
7  I = trapezoid(y, dx=2)
8
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

```
• (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for_lectures2.py
```

21.8

Интегрирование фиксированного набора данных (запись промежуточных результатов интегрирования)

```
1  from scipy.integrate import trapezoid, cumulative_trapezoid
2
3  y = [1, 5, 3.2, 4.4]
4
5
6  I1 = trapezoid(y)
7  I2 = cumulative_trapezoid(y, initial=0)
8
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

JUPYTER

```
• (venv) vitalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/venv/bin/py
or/PROJECTS/test/for_lectures2.py
```

```
trapezoid = 10.9
```

```
cumulative_trapezoid = [ 0.  3.  7.1 10.9]
```

Интегрирование дифференциальных уравнений

<u><code>solve_ivp</code></u> (fun, t_span, y0[, method, t_eval, ...])	Solve an initial value problem for a system of ODEs.
<code>RK23</code> (fun, t0, y0, t_bound[, max_step, rtol, ...])	Explicit Runge-Kutta method of order 3(2).
<code>RK45</code> (fun, t0, y0, t_bound[, max_step, rtol, ...])	Explicit Runge-Kutta method of order 5(4).
<code>DOP853</code> (fun, t0, y0, t_bound[, max_step, ...])	Explicit Runge-Kutta method of order 8.
<code>Radau</code> (fun, t0, y0, t_bound[, max_step, ...])	Implicit Runge-Kutta method of Radau IIA family of order 5.
<code>BDF</code> (fun, t0, y0, t_bound[, max_step, rtol, ...])	Implicit method based on backward-differentiation formulas.
<code>LSODA</code> (fun, t0, y0, t_bound[, first_step, ...])	Adams/BDF method with automatic stiffness detection and switching.
<code>OdeSolver</code> (fun, t0, y0, t_bound, vectorized)	Base class for ODE solvers.
<code>DenseOutput</code> (t_old, t)	Base class for local interpolant over step made by an ODE solver.
<code>OdeSolution</code> (ts, interpolants)	Continuous ODE solution.

Интегрирование дифференциальных уравнений

Для программирования ДУ необходимо перевести в нормальную форму Коши!

Производные искомых функций

Параметры математической модели (динамической системы)

$$\begin{cases} \frac{dx}{dt} = (\alpha - \beta y)x, \\ \frac{dy}{dt} = (-\gamma + \delta x)y, \end{cases}$$

Искомые функции

Параметры математической модели (динамической системы)

Модель Лотки — Вольтерры

Интегрирование дифференциальных уравнений

Программирование системы ДУ.

$$\begin{cases} \frac{dx}{dt} = (\alpha - \beta y)x, \\ \frac{dy}{dt} = (-\gamma + \delta x)y, \end{cases}$$

```
def system(t, q):  
    return [  
        (0.1 - 0.4 * q[1])*q[0],  
        (-1 + 2.3 * q[0])*q[0]  
    ]
```

```
def system(t, q):  
    x, y = q  
    alpha, betta, gamma, delta = 1.5, 1, 3, 1  
  
    dx = (alpha - betta * y)*x  
    dy = (-gamma + delta * x)*y  
    return [dx, dy]
```

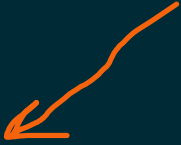
Интегрирование дифференциальных уравнений

Интегрирование системы ДУ.


```
from scipy.integrate import solve_ivp
import numpy as np
```

```
14 # конечное время интегрирования
15 t_end = 20
16 # количество временных точек для интегрирования
17 t = np.linspace(0, t_end, 100)
18 # начальные условия
19 x0, y0 = 10, 5
20 # конфигурирование функции численного интегрирования ДУ
21 ode = solve_ivp(system, [0, t_end], [x0, y0], method='RK23', dense_output=True)
22 # вызов функции численного интегрирования
23 res = ode.sol(t)
24 # получить результаты численного интегрирования ДУ
25 x, y = res[0], res[1]
```

Метод
интегрирования



Разрешить
выдать результат



```
(venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/venv/bin/python /Users/vitalybykador/PROJECTS/test/for_lectures2.py

[10.  0.23954521  0.28808585  0.3642988  0.47387076  0.62635057  0.83551814
 1.12039896  1.50689067  2.03009  2.73708363  3.69148672  4.97617311
 6.6930621  8.93258477 11.49986376 11.69850223 4.39400723 0.98463034
 0.37827259 0.24223728 0.21335045 0.22608759 0.26614501 0.33253271
 0.42967497 0.56580163 0.75314091 1.00874759 1.35588715 1.82610477
 2.46193037 3.32071155 4.47835815 6.03117538 8.08402105 10.62706757
12.44380726 7.29208155 1.61304313 0.49232755 0.26753084 0.21369959
 0.21415913 0.24427089 0.29983554 0.38359837 0.50232183 0.66654332
 0.89117975 1.19669657 1.61088841 2.1713611 2.92868037 3.95075377
 5.32567311 7.15949119 9.52890372 12.0539818 10.77443738 3.15073783
 0.74818016 0.32245775 0.22321259 0.20593356 0.22412058 0.26794575
 0.3377638 0.43863625 0.57932861 0.77253816 1.03582122 1.39320691
 1.87724549 2.53172714 3.41576699 4.60738734 6.20569665 8.31667464
10.91132131 12.54139997 6.66920477 1.4184618 0.44847702 0.25148827
 0.20495158 0.20785467 0.23877253 0.29429964 0.37743435 0.4949956
 0.65742599 0.87950324 1.18150854 1.59099596]

[5.  9.31079875  6.80213776  4.07749343  2.34677052  1.3373918
 0.76212415  0.4371223  0.25403402  0.15042479  0.09145567  0.05769003
```


Интегрирование дифференциальных уравнений

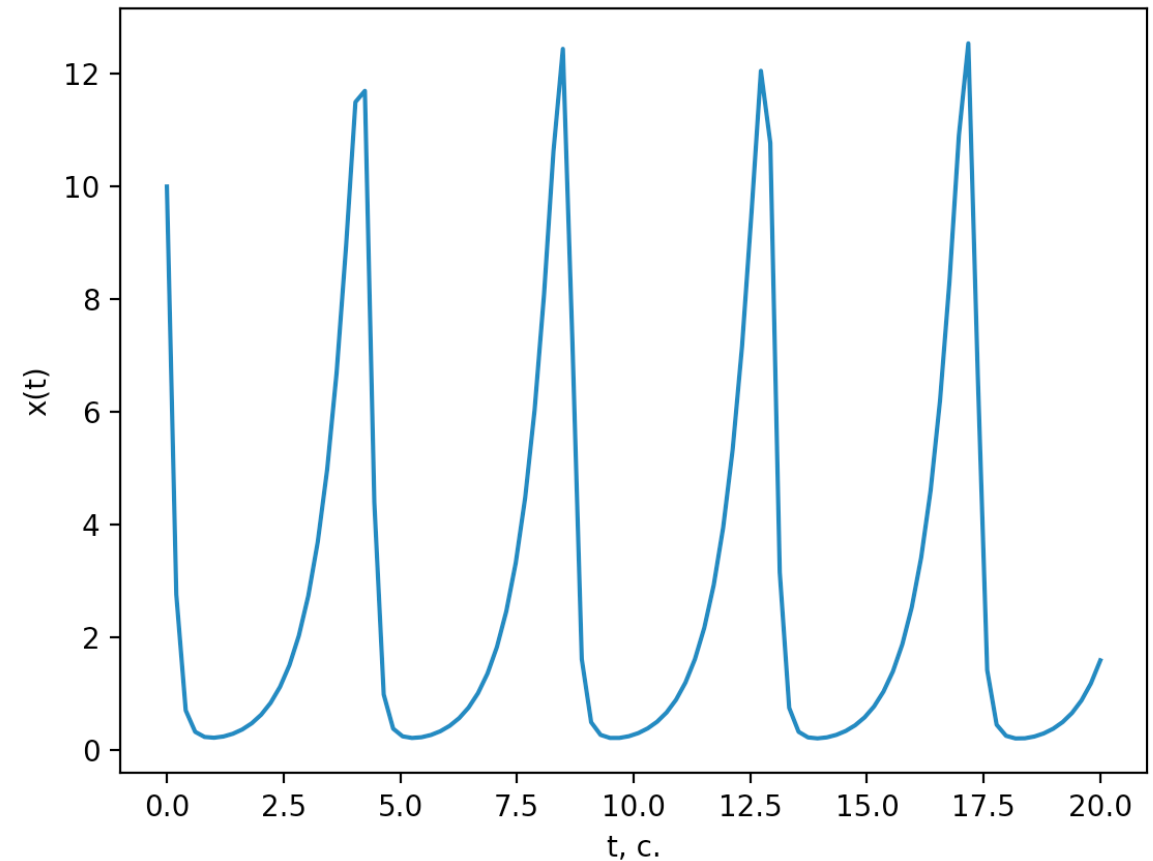
Интегрирование системы ДУ.

```
1  from scipy.integrate import solve_ivp
2  import numpy as np
3
4
5
6  def system(t, q, alpha, betta, gamma, delta):
7      x, y = q
8
9      dx = (alpha - betta * y)*x
10     dy = (-gamma + delta * x)*y
11     return [dx, dy]
12
13     t_end = 20
14     t = np.linspace(0, t_end, 100)
15     x0, y0 = 10, 5
16     ode = solve_ivp(system, [0, t_end], [x0, y0], method='RK23', args=(1.5, 1, 3, 1), dense_output=True)
17     res = ode.sol(t)
18     x, y = res[0], res[1]
```

Графики решения дифференциальных уравнений

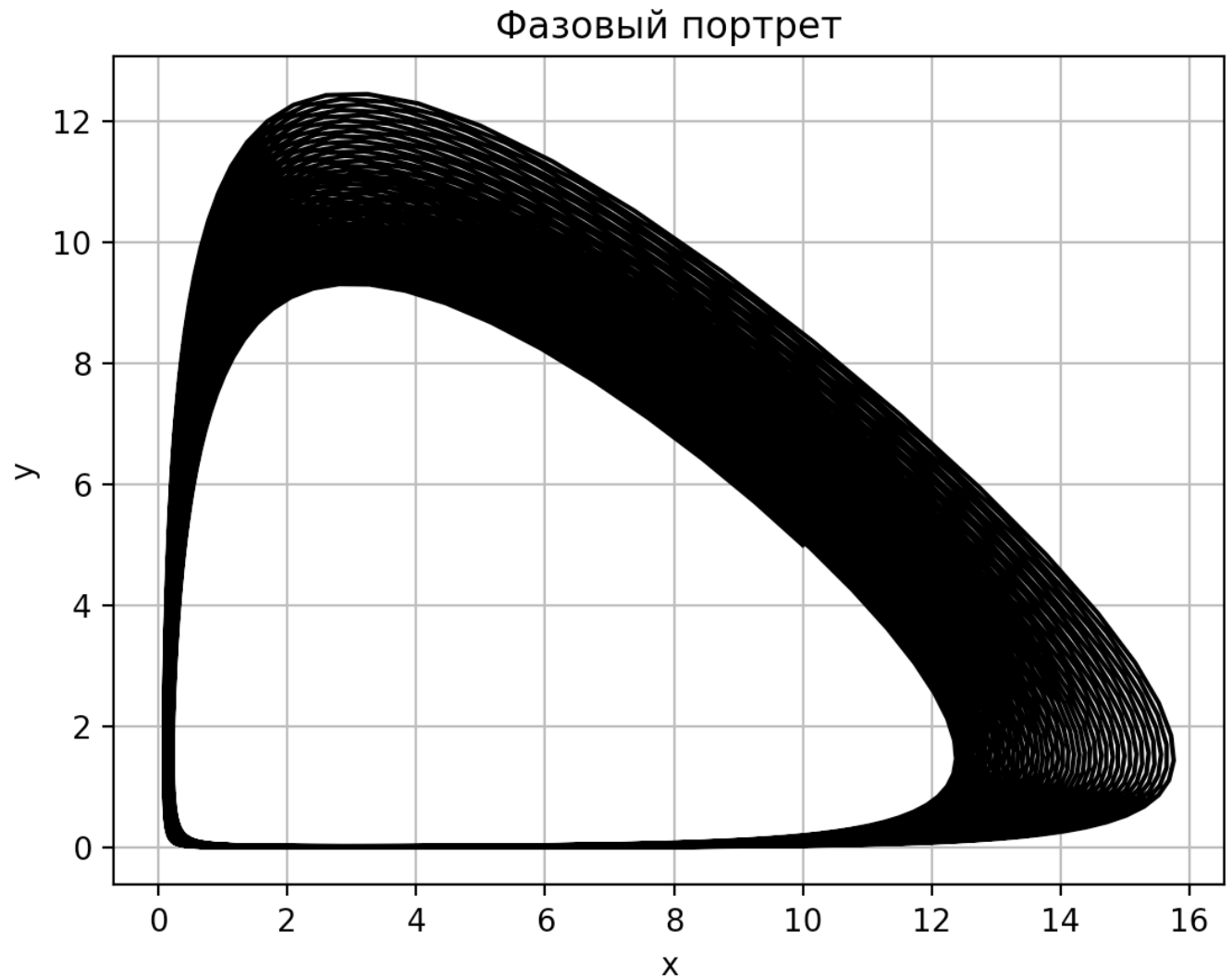
```
1 from scipy.integrate import solve_ivp
2 import numpy as np
3 import matplotlib.pyplot as plt
```

```
17 ode = solve_ivp(system,
18   res = ode.sol(t)
19   x, y = res[0], res[1]
20
21   pt.plot(t, x)
22   pt.xlabel('t, c.')
23   pt.ylabel('x(t)')
24   pt.show()
25
```



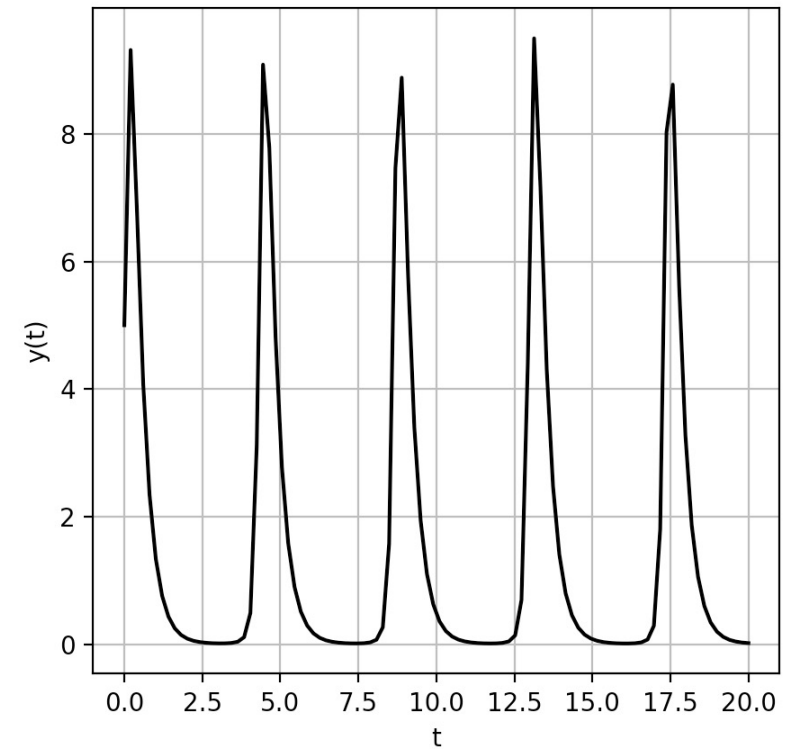
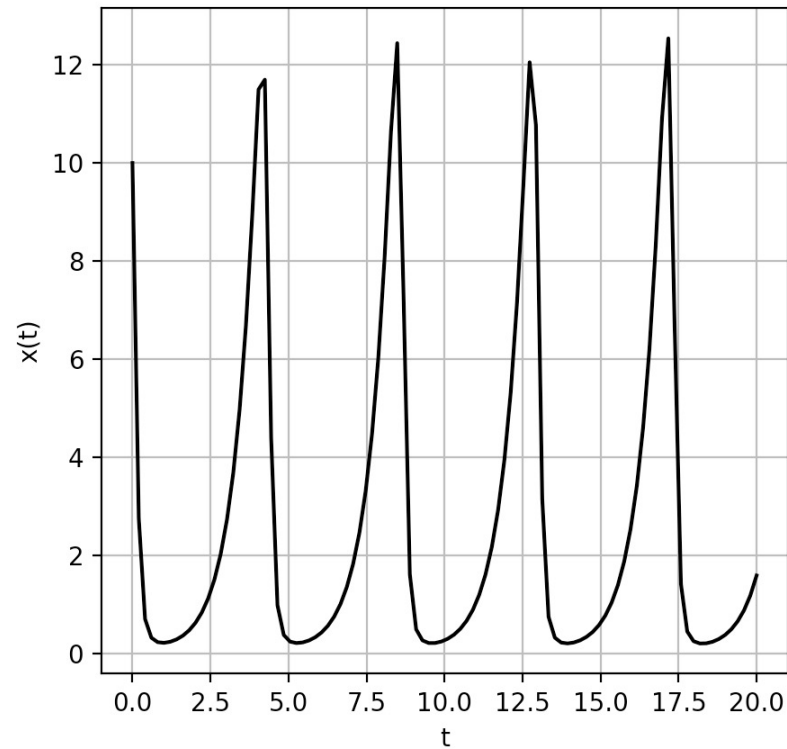
Графики решения дифференциальных уравнений

```
20  
21 pt.plot(x, y, '-k')  
22 pt.title('Фазовый портрет')  
23 pt.xlabel('x')  
24 pt.ylabel('y')  
25 pt.grid()  
26 pt.show()  
27
```



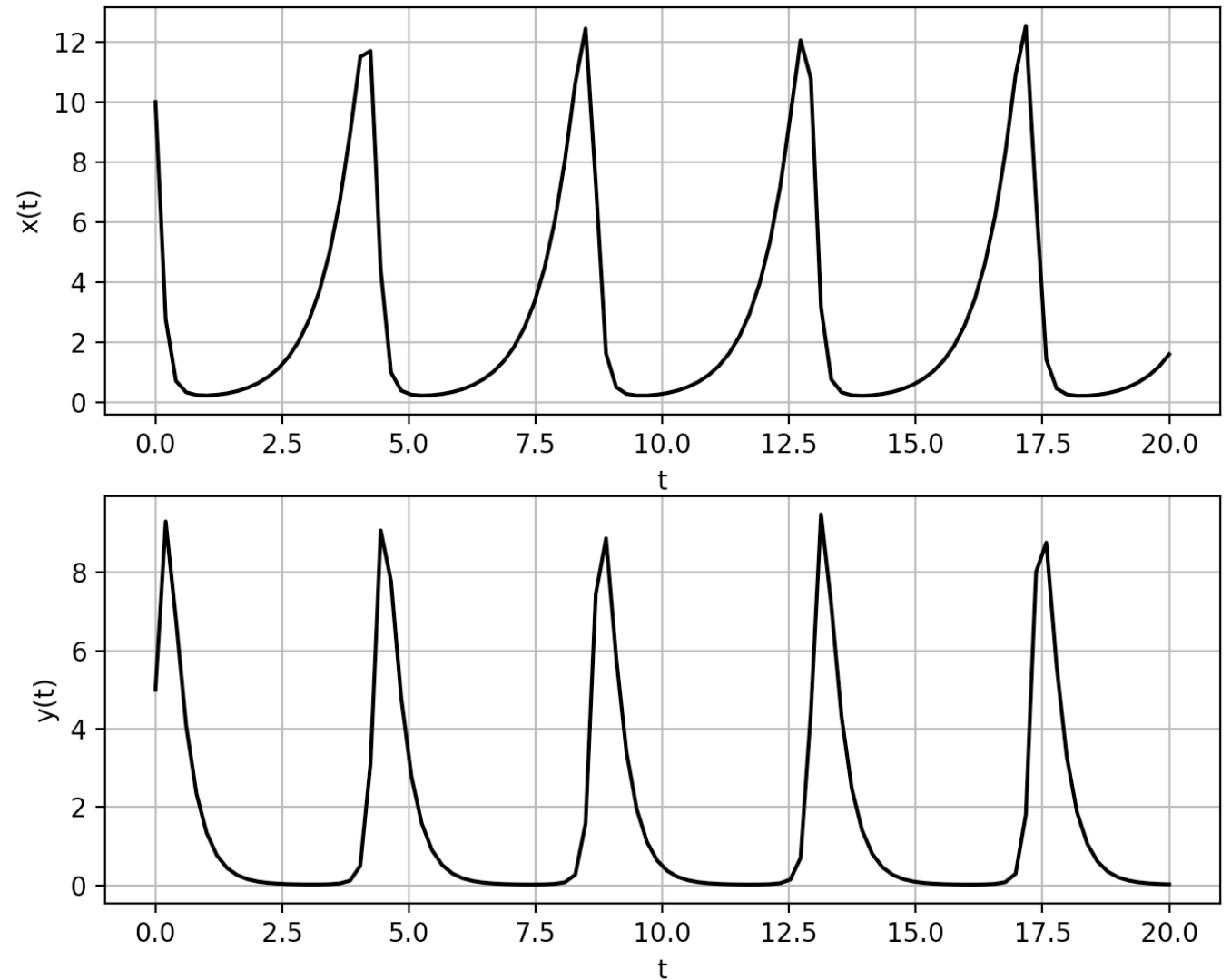
Графики решения дифференциальных уравнений

```
20
21 pt.subplot(1,2,1)
22 pt.plot(t, x, '-k')
23 pt.xlabel('t')
24 pt.ylabel('x(t)')
25 pt.grid()
26
27 pt.subplot(1,2,2)
28 pt.plot(t, y, '-k')
29 pt.xlabel('t')
30 pt.ylabel('y(t)')
31 pt.grid()
32
33 pt.show()
```



Графики решения дифференциальных уравнений

```
20 |
21 pt.subplot(2,1,1)
22 pt.plot(t, x, '-k')
23 pt.xlabel('t')
24 pt.ylabel('x(t)')
25 pt.grid()
26
27 pt.subplot(2,1,2)
28 pt.plot(t, y, '-k')
29 pt.xlabel('t')
30 pt.ylabel('y(t)')
31 pt.grid()
32
33 pt.show()
```

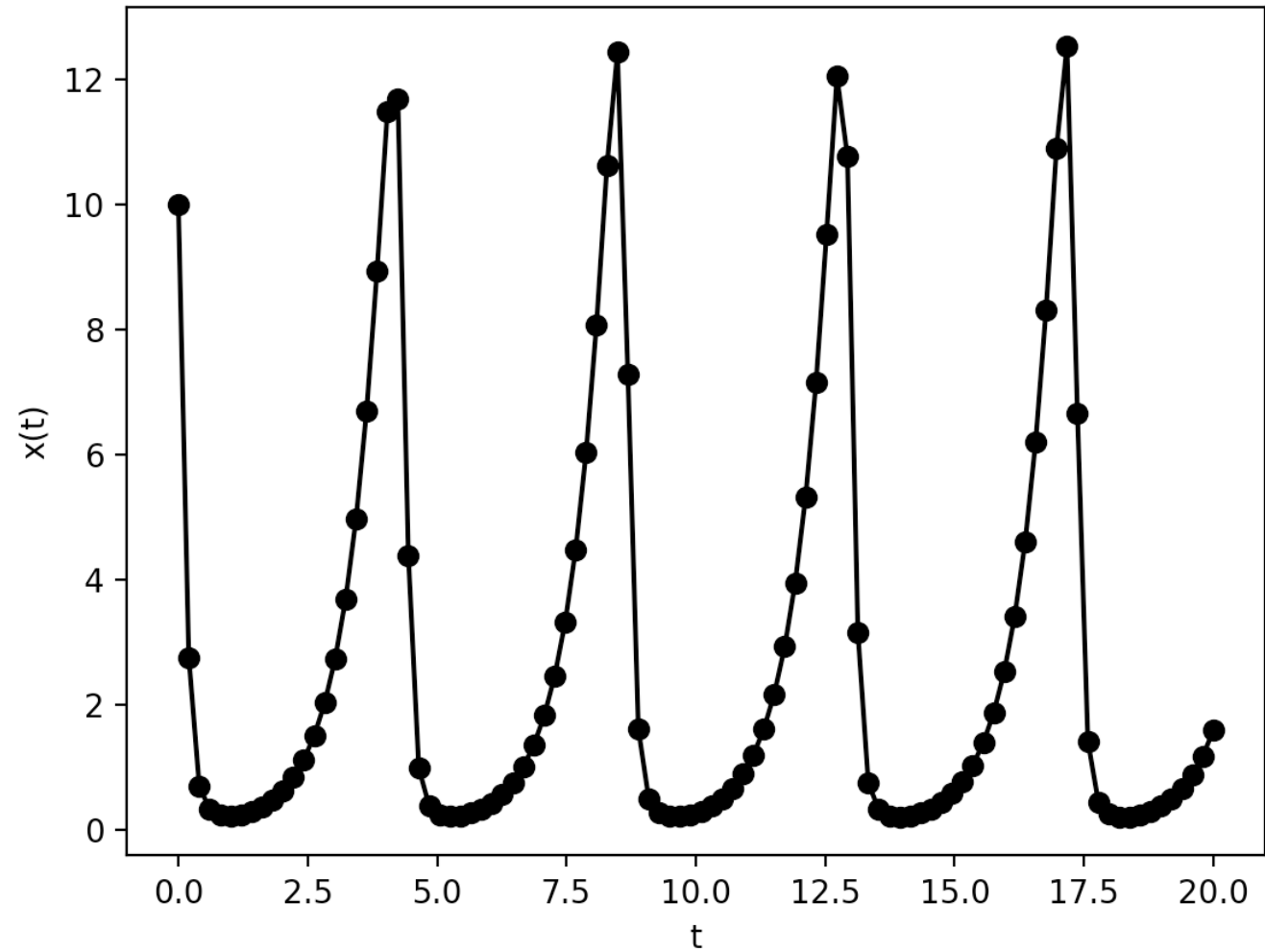


Графики решения дифференциальных уравнений

Линия сплошная Маркер в виде кружка

```
21 pt.plot(t, x, '-ok')
22 pt.xlabel('t')
23 pt.ylabel('x(t)')
24
25 pt.show()
```

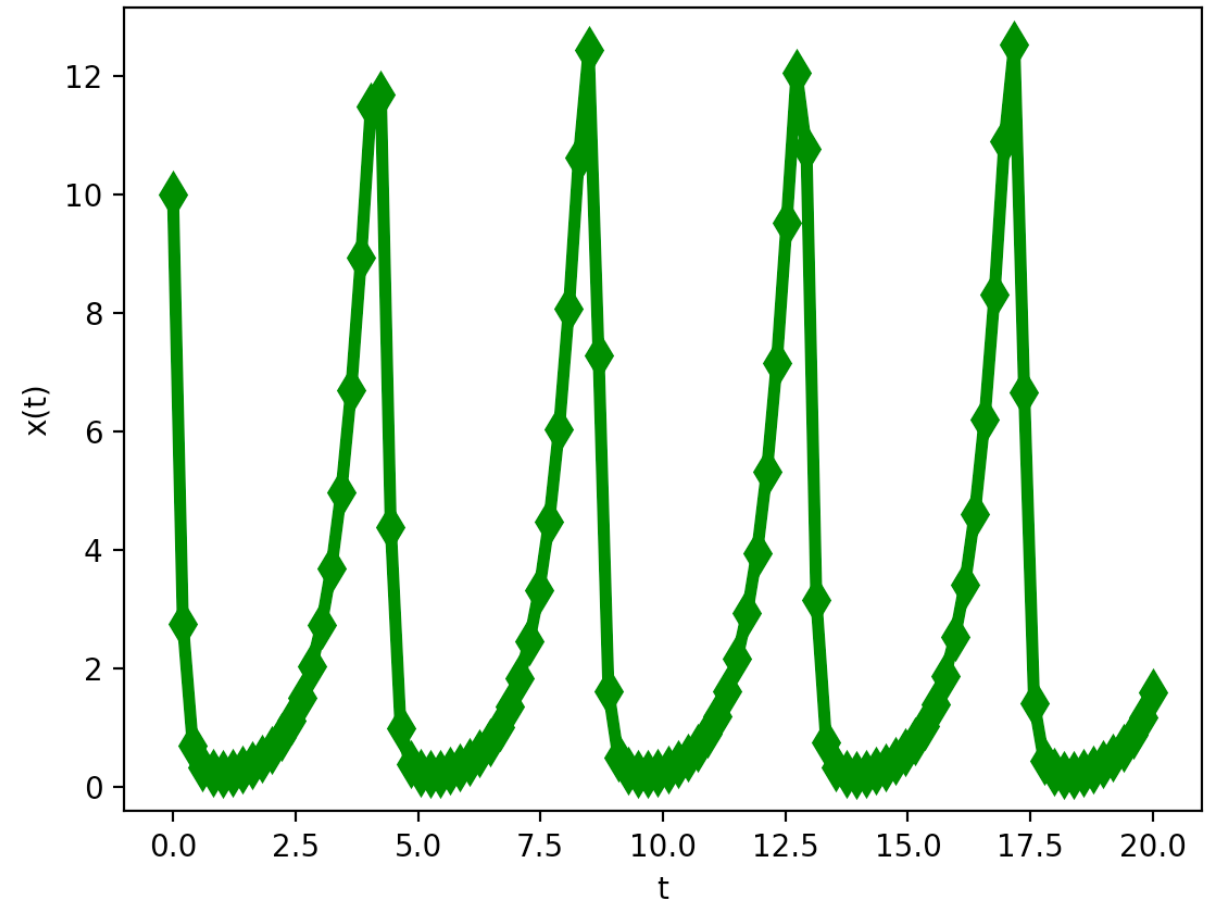
Цвет чёрный



Графики решения дифференциальных уравнений

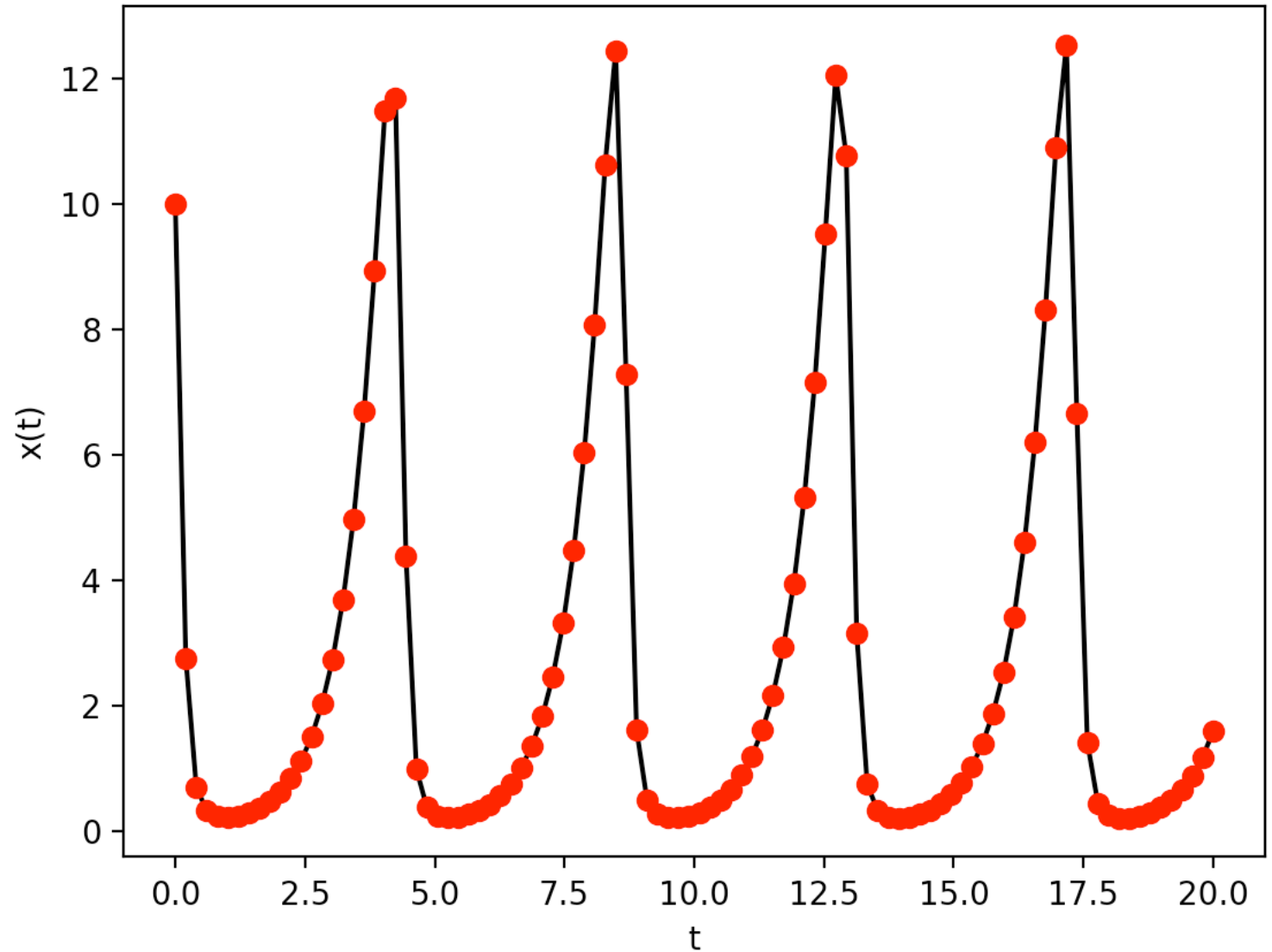
```
plt.plot(t, x, '-dg', ms=10, lw=4)
```

Линия сплошная, маркер в виде ромбика, всё зелёного цвета.
`ms=10` – это размер маркера (**m**arker **s**ize), то есть «ромбика».
`lw=4` - это толщина линии (**l**ine **w**idth).



Графики решения дифференциальных уравнений

```
21 pt.plot(t, x, '-k')
22 pt.plot(t, x, 'or')
23 pt.xlabel('t')
24 pt.ylabel('x(t)')
25
26 pt.show()
27
```



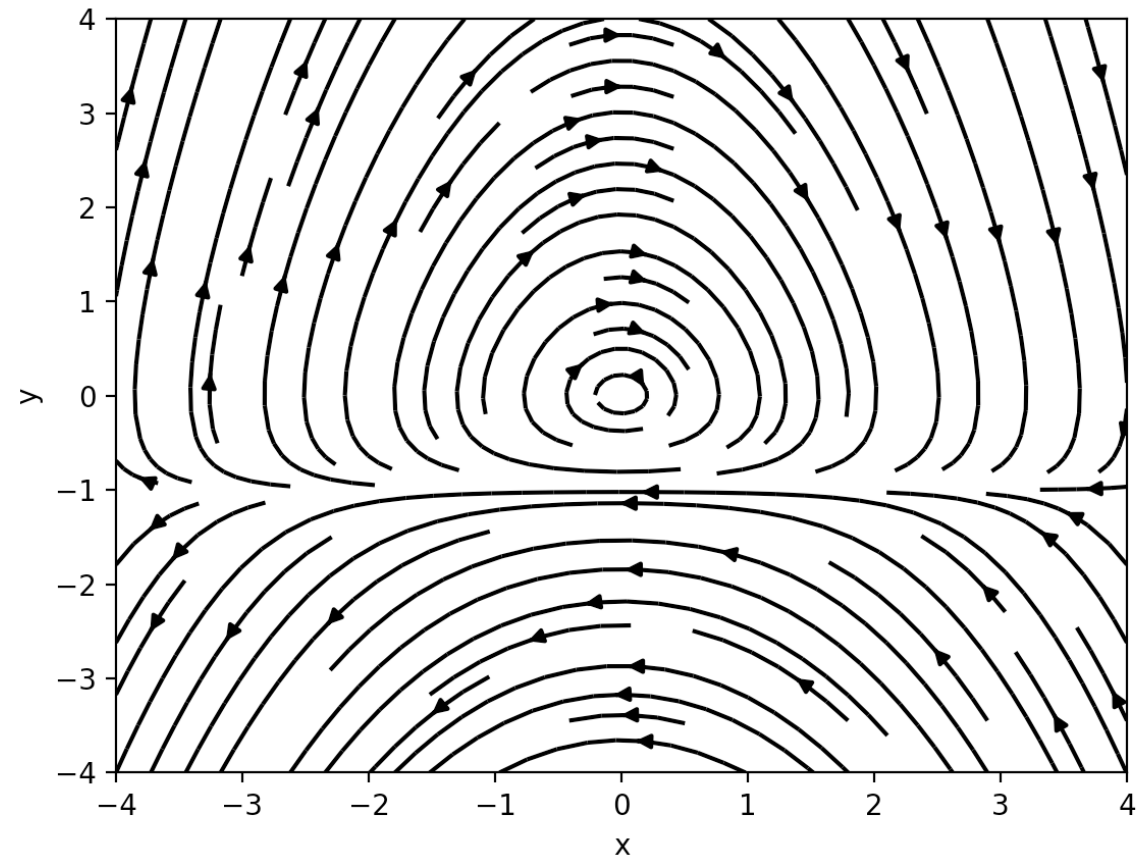
$$\ddot{x} + x \cdot \dot{x} + x = 0 \quad \rightarrow \quad \begin{cases} \frac{dx}{dt} = y \\ \frac{dy}{dt} = -x \cdot y - x \end{cases}$$

Решение ДУ без
численного
интегрирования
(только для ДУ 2-го
порядка)

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3
4
5  x = np.linspace(-4, 4, 10)
6  y = x
7  X, Y = np.meshgrid(x, y)
8
9  DX = Y
10 DY = -X*Y - X
11
12 plt.streamplot(X, Y, DX, DY, color='k')
13
14 plt.xlabel('x')
15 plt.ylabel('y')
16 plt.show()

```



Быстрое преобразование Фурье

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft

dt = 0.00001
t = np.arange(0, 1, dt)
y = 0.2*np.sin(2*np.pi*50*t) + 0.1*np.cos(2*np.pi*150*t) + 0.25*np.sin(2*np.pi*100*t) + np.random.randn(len(t))

N = len(t)
Y = fft(y)

A = abs(2*Y/N)

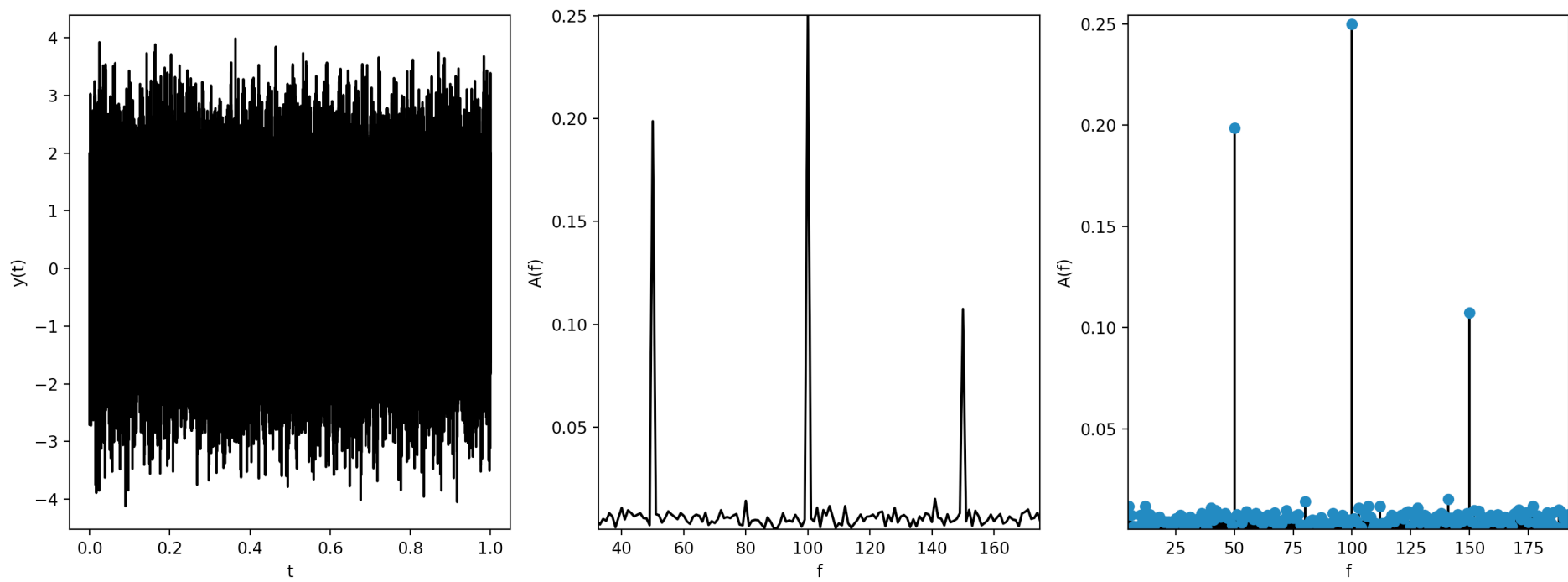
fs = 1/dt
m = np.array(range(N))
f = m*fs/N
```

Быстрое преобразование Фурье

```
pt.subplot(1,3,1)  
pt.plot(t, y, 'k')  
pt.xlabel('t')  
pt.ylabel('y(t)')
```

```
pt.subplot(1,3,2)  
pt.plot(f, A, 'k')  
pt.xlabel('f')  
pt.ylabel('A(f)')  
pt.subplot(1,3,3)
```

```
pt.stem(f, A, 'k')  
pt.xlabel('f')  
pt.ylabel('A(f)')  
pt.show()
```





Q Search the docs ...

- Clustering package (`scipy.cluster`)
- K-means clustering and vector quantization (`scipy.cluster.vq`)
- Hierarchical clustering (`scipy.cluster.hierarchy`)
- Constants (`scipy.constants`)
- Discrete Fourier transforms (`scipy.fft`)
- Legacy discrete Fourier transforms (`scipy.fftpack`)
- Integration and ODEs (`scipy.integrate`)
- Interpolation (`scipy.interpolate`)
- Input and output (`scipy.io`)
- Linear algebra (`scipy.linalg`)
- Low-level BLAS functions (`scipy.linalg.blas`)
- Low-level LAPACK functions (`scipy.linalg.lapack`)
- BLAS Functions for Cython
- LAPACK functions for Cython
- Interpolative matrix decomposition (`scipy.linalg.interpolative`)
- Miscellaneous routines (`scipy.misc`)
- Multidimensional image processing (`scipy.ndimage`)
- Orthogonal distance regression (`scipy.odr`)

- Optimization and root finding (`scipy.optimize`)
- Cython optimize zeros API
- Signal processing (`scipy.signal`)
- Sparse matrices (`scipy.sparse`)
- Sparse linear algebra (`scipy.sparse.linalg`)
- Compressed sparse graph routines (`scipy.sparse.csgraph`)
- Spatial algorithms and data structures (`scipy.spatial`)
- Distance computations (`scipy.spatial.distance`)
- Special functions (`scipy.special`)
- Statistical functions (`scipy.stats`)
- Result classes
- Contingency table functions (`scipy.stats.contingency`)
- Statistical functions for masked arrays (`scipy.stats.mstats`)
- Quasi-Monte Carlo submodule (`scipy.stats.qmc`)
- Random Number Generators (`scipy.stats.sampling`)
- Low-level callback functions

Linear algebra (`scipy.linalg`)

Linear algebra functions.

See also

`numpy.linalg` for more linear algebra functions. Note that although `scipy.linalg` imports most of them, identically named functions from `scipy.linalg` may offer more or slightly differing functionality.

Basics

<code>inv(a[, overwrite_a, check_finite])</code>	Compute the inverse of a matrix.
<code>solve(a, b[, sym_pos, lower, overwrite_a, ...])</code>	Solves the linear equation set $a \cdot x = b$ for the unknown x for square a matrix.
<code>solve_banded(l_and_u, ab, b[, overwrite_ab, ...])</code>	Solve the equation $a \cdot x = b$ for x , assuming a is banded matrix.
<code>solveh_banded(ab, b[, overwrite_ab, ...])</code>	Solve equation $a \cdot x = b$.
<code>solve_circulant(c, b[, singular, tol, ...])</code>	Solve $C \cdot x = b$ for x , where C is a circulant matrix.
<code>solve_triangular(a, b[, trans, lower, ...])</code>	Solve the equation $a \cdot x = b$ for x , assuming a is a triangular matrix.
<code>solve_toeplitz(c_or_cr, b[, check_finite])</code>	Solve a Toeplitz system using Levinson Recursion
<code>matmul_toeplitz(c_or_cr, x[, check_finite, ...])</code>	Efficient Toeplitz Matrix-Matrix Multiplication using FFT
<code>det(a[, overwrite_a, check_finite])</code>	Compute the determinant of a matrix

<code>norm(a[, ord, axis, keepdims, check_finite])</code>	Matrix or vector norm.
<code>lstsq(a, b[, cond, overwrite_a, ...])</code>	Compute least-squares solution to equation $Ax = b$.
<code>pinv(a[, atol, rtol, return_rank, ...])</code>	Compute the (Moore-Penrose) pseudo-inverse of a matrix.
<code>pinvh(a[, atol, rtol, lower, return_rank, ...])</code>	Compute the (Moore-Penrose) pseudo-inverse of a Hermitian matrix.
<code>kron(a, b)</code>	Kronecker product.
<code>khatrao(a, b)</code>	Khatri-rao product
<code>tril(m[, k])</code>	Make a copy of a matrix with elements above the k th diagonal zeroed.
<code>triu(m[, k])</code>	Make a copy of a matrix with elements below the k th diagonal zeroed.
<code>orthogonal_procrustes(A, B[, check_finite])</code>	Compute the matrix solution of the orthogonal Procrustes problem.
<code>matrix_balance(A[, permute, scale, ...])</code>	Compute a diagonal similarity transformation for row/column balancing.
<code>subspace_angles(A, B)</code>	Compute the subspace angles between two matrices.
<code>bandwidth(a)</code>	Return the lower and upper bandwidth of a 2D numeric array.

И другие функции.