Handouts/Lecture 4 (Calculus and graphics)/Lecture 4: Graphics/Graphics.sagews

Author	Eugene Strakhov
Date	2019-06-29T19:07:21
Project	07c06dbe-4967-451f-aa68-dd9268bd2ece
Location	Handouts/Lecture 4 (Calculus and graphics)/Lecture 4: Graphics/Graphics.sagews
Original file	Graphics.sagews

Графика в Sage

Графика в Sage

- 2D-графики и графические примитивы (via matplotlib)
- 3D-графики и графические примитивы
- Свойства и методы объекта Graphics()
- Полный список опций для 2D-графиков: http://doc.sagemath.org/html/en/reference/plotting/sage/plot/plot.html#sage.plot.plot.plot

Все двумерные графические объекты в SageMath принадлежат классу `Graphics`. Массивы двумерных графиков принадлежат классу `GraphicsArray`.

Создать **пустой графический объект** можно так:

```
p = Graphics()
```

Для графических объектов определена **операция «суммирования»**, т. е. отображения двух и более графических объектов на одном рисунке. При этом объекты накладываются друг на друга, как слои в «фотошопе».

Чтобы **отобразить графический объект на экране**, нужно вызвать метод `show()`:

```
p.show(...options...)
```

Чтобы **сохранить картинку во внешний файл**, нужно вызвать метод `save()`:

```
p.save(...options...)
```

Все двумерные графические объекты в SageMath принадлежат классу `Graphics`. Массивы двумерных графиков принадлежат классу `GraphicsArray`.

Создать **пустой графический объект** можно так:

```
p = Graphics()
```

Для графических объектов определена **операция «суммирования»**, т. е. отображения двух и более графических объектов на одном рисунке. При этом объекты накладываются друг на друга, как слои в «фотошопе».

Чтобы **отобразить графический объект на экране**, нужно вызвать метод `show()`:

```
p.show(...options...)
```

Чтобы **сохранить картинку во внешний файл**, нужно вызвать метод `save()`:

```
p.save(...options...)
```

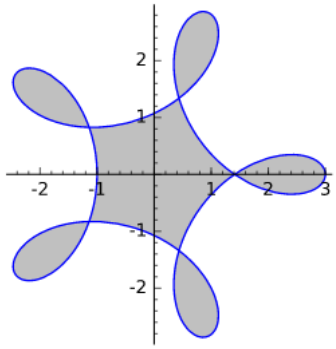
Пример 1. График функции, заданной параметрически:

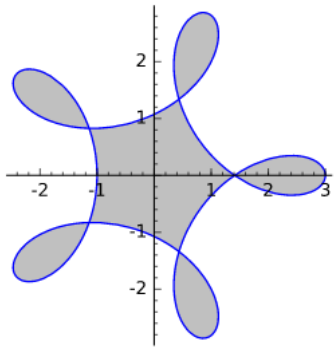
$$\begin{cases} x(t) = \cos t + 2\cos\dfrac{t}{4}, \\ y(t) = \sin t - 2\sin\dfrac{t}{4}, \quad t \in [0; 8\pi]. \end{cases}$$

Пример 1. График функции, заданной параметрически:

$$\begin{cases} x(t) = \cos t + 2\cos\dfrac{t}{4}, \\ y(t) = \sin t - 2\sin\dfrac{t}{4}, \quad t \in [0; 8\pi]. \end{cases}$$

```
t = var('t')
p = parametric_plot([cos(t) + 2*cos(t/4), sin(t) - 2*sin(t/4)], (t, 0, 8*pi), fill = True)
p.show(figsize = [4, 3]) # длина и ширина - в дюймах
p.save('hypotrochoid.png', figsize = [10, 8], dpi = 300) # огромный рисунок с высоким разрешением
```

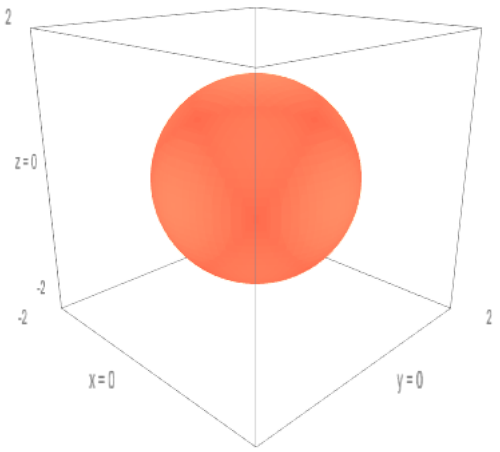




Пример 2. Сфера как график неявной функции.

Пример 2. Сфера как график неявной функции.

```
x, y, z = var('x y z')
f = x^2 + y^2 + z^2 == 2
ip = implicit_plot3d(f, (x, -2, 2), (y, -2, 2), (z, -2, 2), color = 'tomato')
ip.show(aspect_ratio=[1,1,1]) # равные масштабы по осям
```




```
# Все ключевые слова для обозначения цвета
colors.keys()
```

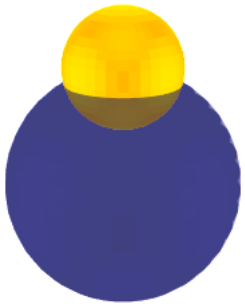
```
['indigo', 'gold', 'firebrick', 'indianred', 'yellow', 'darkolivegreen', 'darkseagreen', 'slategrey', 'darkslategrey', 'mediumvioletred',
'mediumorchid', 'chartreuse', 'mediumslateblue', 'black', 'springgreen', 'crimson', 'lightsalmon', 'brown', 'turquoise', 'olivedrab', 'lightcy
'cyan', 'silver', 'skyblue', 'gray', 'darkturquoise', 'goldenrod', 'darkgreen', 'darkviolet', 'darkgray', 'lightpink', 'teal', 'darkmagenta',
'automatic', 'lightgoldenrodyellow', 'lavender', 'yellowgreen', 'thistle', 'violet', 'navy', 'dimgrey', 'orchid', 'blue', 'ghostwhite',
'honeydew', 'cornflowerblue', 'darkblue', 'darkkhaki', 'mediumpurple', 'cornsilk', 'red', 'bisque', 'slategray', 'darkcyan', 'khaki', 'wheat'
'deepskyblue', 'darkred', 'steelblue', 'aliceblue', 'lightslategrey', 'gainsboro', 'mediumturquoise', 'floralwhite', 'plum', 'purple',
'lightgrey', 'burlywood', 'darksalmon', 'beige', 'azure', 'lightsteelblue', 'oldlace', 'greenyellow', 'royalblue', 'lightseagreen', 'sienna',
'lightcoral', 'orangered', 'navajowhite', 'lime', 'palegreen', 'mistyrose', 'seashell', 'mediumspringgreen', 'fuchsia', 'papayawhip',
'blanchedalmond', 'peru', 'aquamarine', 'white', 'darkslategray', 'lightgray', 'ivory', 'dodgerblue', 'lawngreen', 'chocolate', 'orange',
'forestgreen', 'darkgrey', 'olive', 'mintcream', 'antiquewhite', 'darkorange', 'cadetblue', 'moccasin', 'limegreen', 'saddlebrown', 'grey',
'darkslateblue', 'lightskyblue', 'deeppink', 'coral', 'aqua', 'darkgoldenrod', 'maroon', 'sandybrown', 'magenta', 'tan', 'rosybrown', 'pink',
'lightblue', 'palevioletred', 'mediumseagreen', 'slateblue', 'dimgray', 'powderblue', 'seagreen', 'snow', 'mediumblue', 'midnightblue',
```

```
'paleturquoise', 'palegoldenrod', 'whitesmoke', 'darkorchid', 'salmon', 'lightslategray', 'lemonchiffon', 'lightgreen', 'tomato', 'hotpink',
'lightyellow', 'lavenderblush', 'linen', 'mediumaquamarine', 'green', 'blueviolet', 'peachpuff']
```

Пример 3. Сфера как графический примитив.

Пример 3. Сфера как графический примитив.

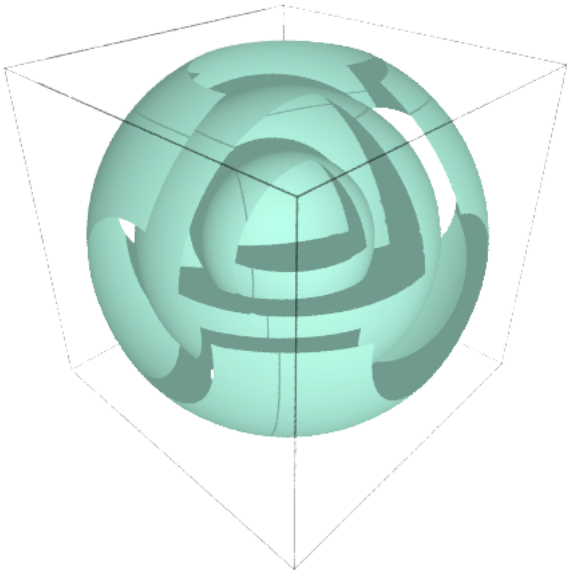
```
two_spheres = sphere(color='orange') + sphere(color=(0,0,0.3), center=(0,0,-2), size=2, opacity=0.5)
two_spheres.show(frame=False, aspect_ratio=[1,1,1]) # сплющенный, т. к. разные масштабы по осям
```

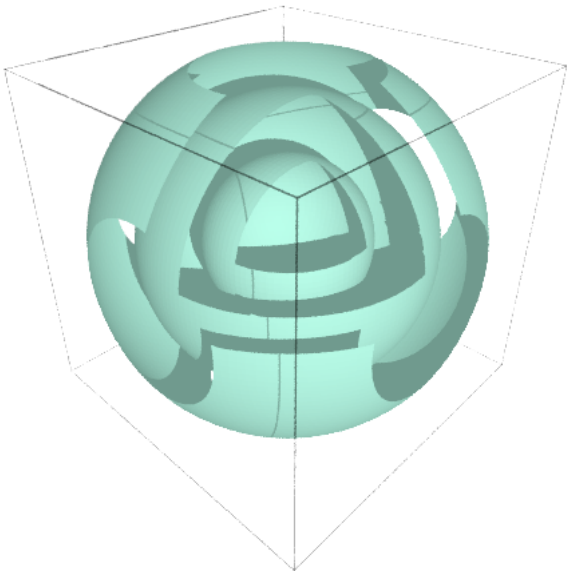


Пример 4. Шар с вырезанными областями.

Пример 4. Шар с вырезанными областями.

```
implicit_plot3d((x^2 + y^2 + z^2), (x,-2,2), (y,-2,2), (z,-2,2), \
                plot_points=60, contour=[1,3,5], \
                region=lambda x,y,z: x<=0.2 or y>=0.2 or z<=0.2, \
                color='aquamarine').show(viewer='tachyon')
```

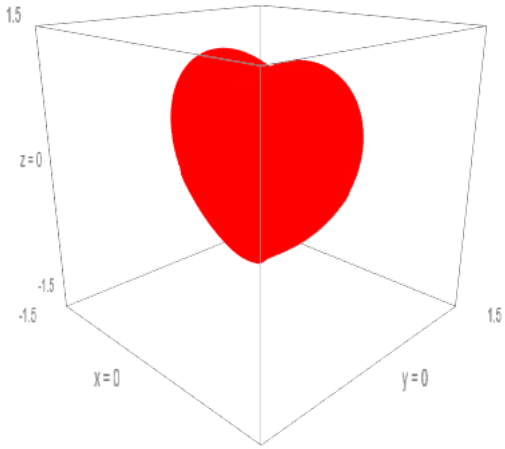




Пример 5. Сердце как трёхмерный график неявной функции.

Пример 5. Сердце как трёхмерный график неявной функции.

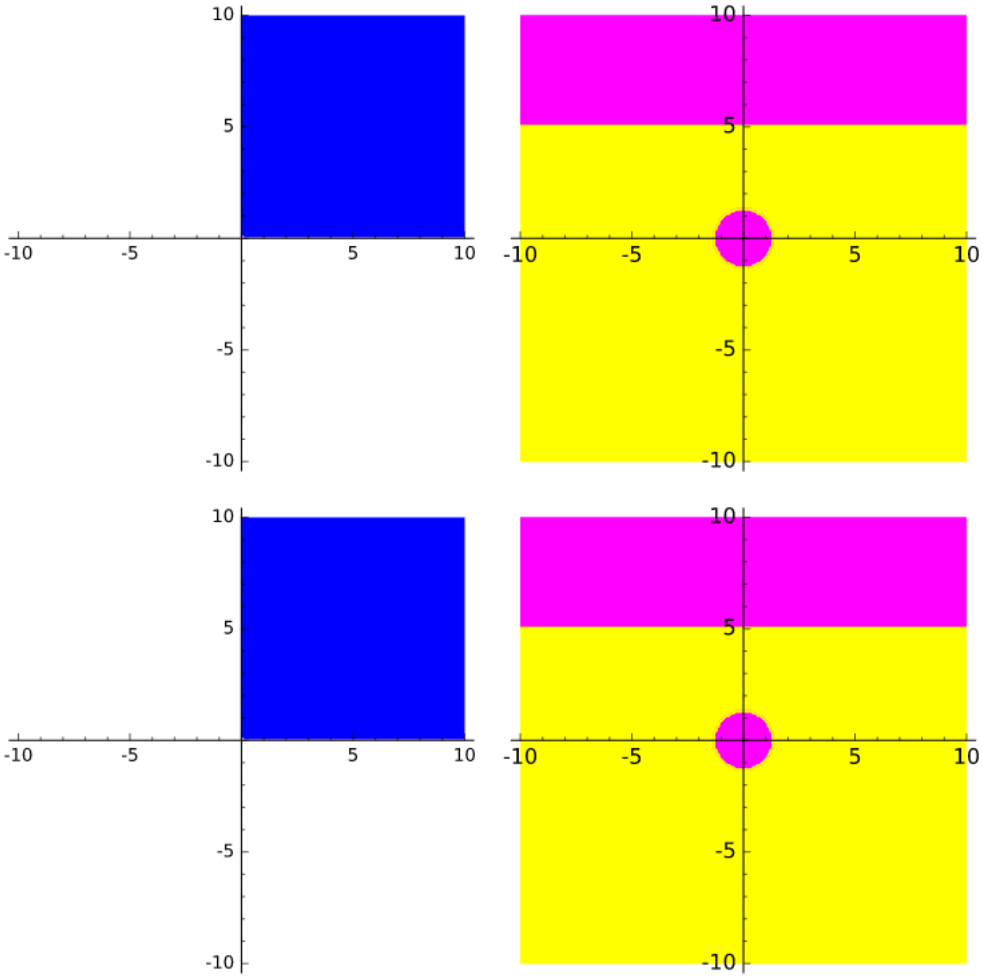
```
F = (x^2+9/4*y^2+z^2-1)^3 - x^2*z^3 - 9/(80)*y^2*z^3
r = 1.5
implicit_plot3d(F, (x,-r,r), (y,-r,r), (z,-r,r), plot_points=80, color='red', smooth=False).show()
```



Пример 6. Графики неравенств (областей).

Пример 6. Графики неравенств (областей).

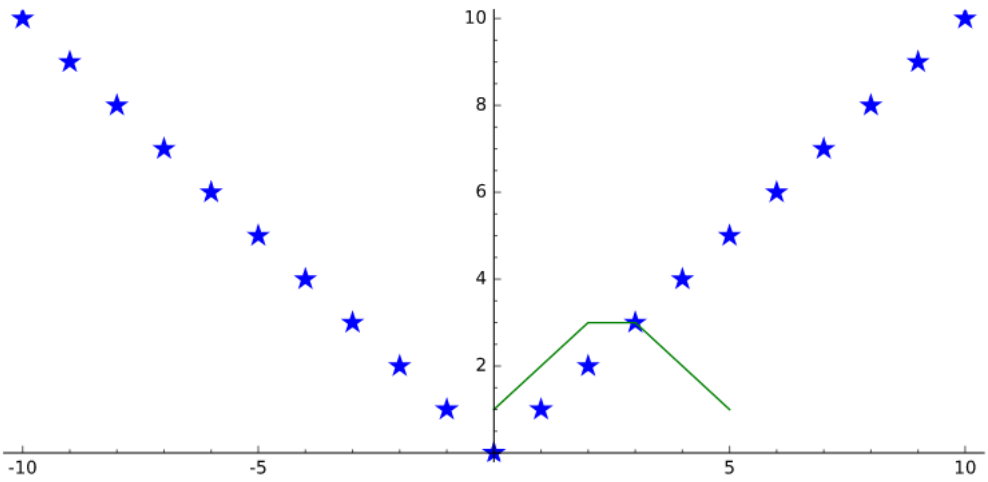
```
y = var('y')
rp1 = region_plot([x>=0, y>=0], (x, -10, 10), (y, -10, 10))
f = lambda x, y : x^2 + y^2 <= sqrt(3) or y >= 5
rp2 = region_plot(f, (x, -10, 10), (y, -10, 10), incol = 'magenta', outcol = 'yellow', plot_points=200)
garr = graphics_array([rp1, rp2]) # массив графиков
garr.show()
```

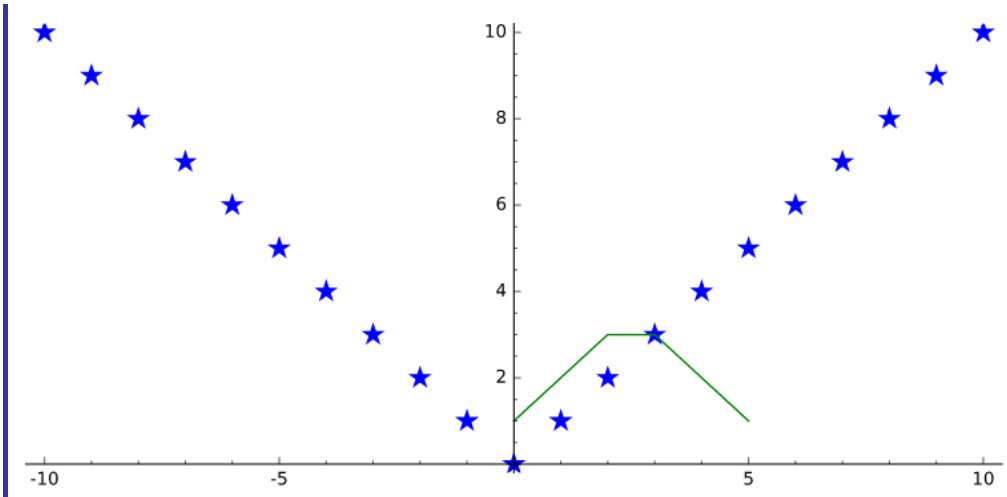


Пример 7. Графики списков.

Пример 7. Графики списков.

```
26 L1 = [(x, abs(x)) for x in range(-10, 11)]
27 lp1 = list_plot(L1, marker='*', size = 200)
28 L2 = [1, 2, 3, 3, 2, 1]
29 lp2 = list_plot(L2, color = 'green',  plotjoined = True)
30 (lp1 + lp2).show()
```



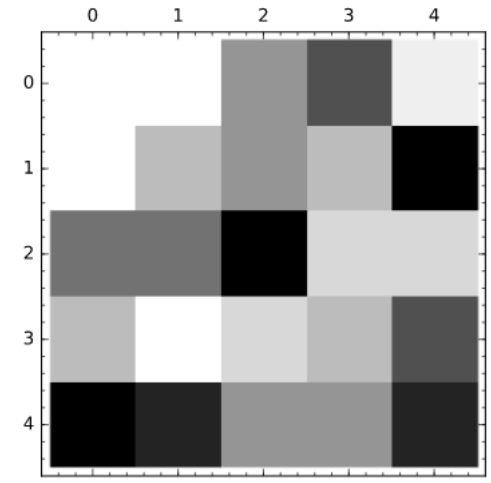


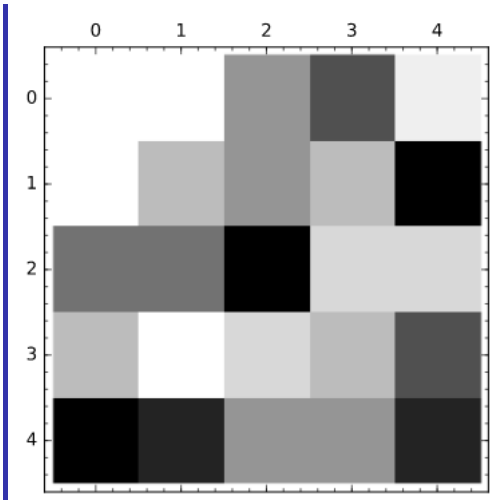
Пример 8. Графики матриц.

Пример 8. Графики матриц.

```
M = random_matrix(ZZ, 5, x=0, y=9)
show(M)
mp = matrix_plot(M) # оттенки серого: чем светлее клетка, тем больше элемент матрицы
show(mp)
```

$$\begin{pmatrix} 0 & 0 & 4 & 6 & 1 \\ 0 & 3 & 4 & 3 & 8 \\ 5 & 5 & 8 & 2 & 2 \\ 3 & 0 & 2 & 3 & 6 \\ 8 & 7 & 4 & 4 & 7 \end{pmatrix}$$

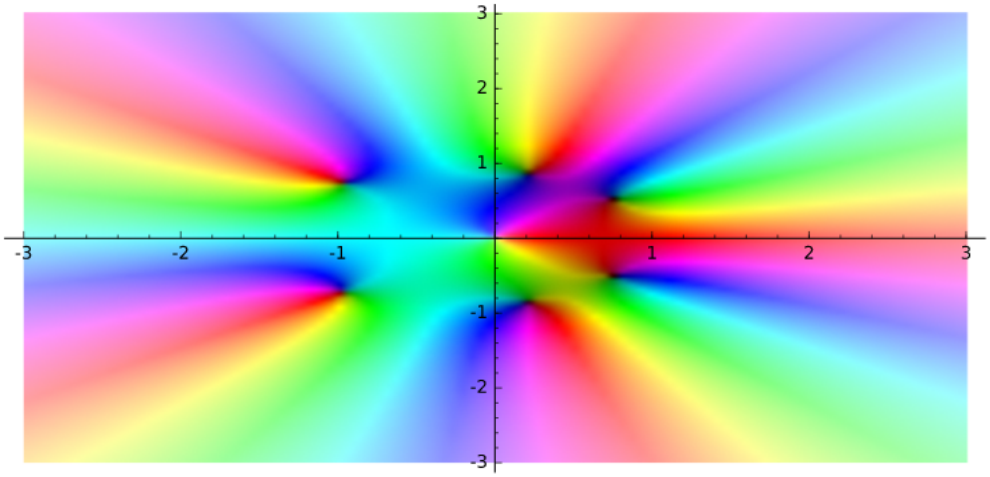


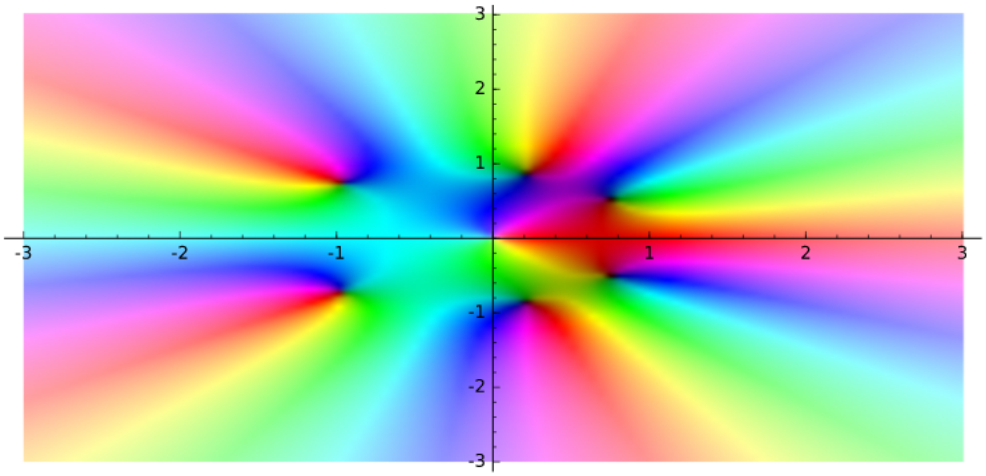


Пример 9. График функции комплексной переменной.

Пример 9. График функции комплексной переменной.

```
f(z) = z^5 + z - 1 + 1/z
complex_plot(f, (-3, 3), (-3, 3))
# Модуль - яркость (brightness), аргумент - цветовой тон (hue), см. модель HSV (HSB): https://ru.wikipedia.org/wiki/HSV_%28%D1%86%D0%B2%D0%B5
```





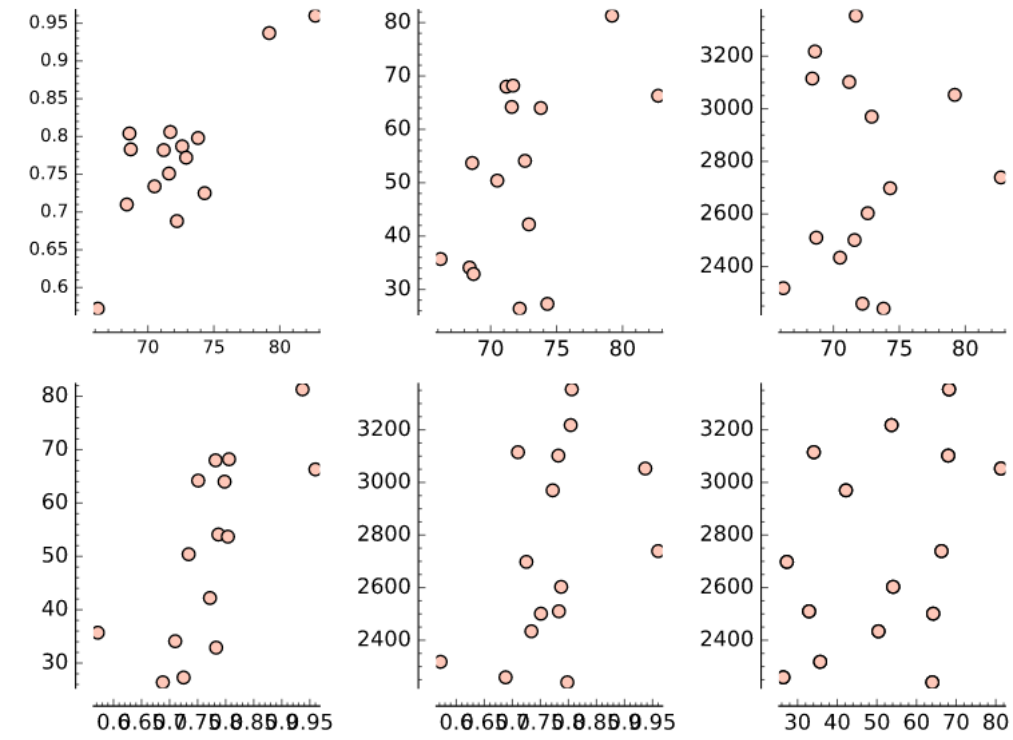
Пример 10. Массив графиков.

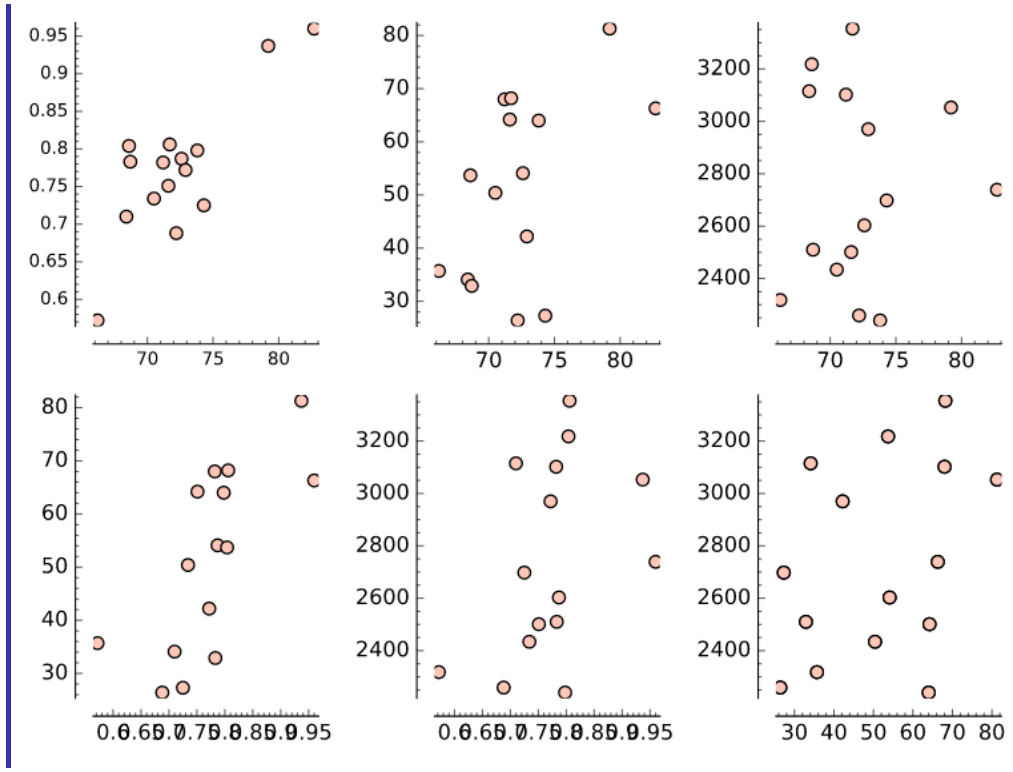
Пример 10. Массив графиков.

```
# Построим попарные диаграммы рассеяния для данных из файла countries.csv
f = open('countries.csv', 'r')
header = f.readline()[:-1].split(';') # заголовок таблицы
data = []
for line in f :
    row = line[:-1].replace(',', '.').split(';')[1:] # подумайте над этой конструкцией
    data.append([float(x) for x in row])
f.close()
data = matrix(data)
show('Матрица исходных данных:', data)
# Теперь построим всевозможные диаграммы рассеяния
ga = [] # заготовка для массива графиков
for i in xrange(data.ncols()-1) :
    for j in range(i+1, data.ncols()) :
        x = data.column(i)
        y = data.column(j)
        ga.append(scatter_plot(zip(x, y)))
graphics_array(ga, nrows=2, ncols=3)
```

Матрица исходных данных:

$$\begin{pmatrix} 72.6 & 0.787 & 54.1 & 2603.0 \\ 73.8 & 0.798 & 64.0 & 2240.0 \\ 74.3 & 0.725 & 27.3 & 2698.0 \\ 70.5 & 0.734 & 50.4 & 2434.0 \\ 71.2 & 0.782 & 68.0 & 3102.0 \\ 68.6 & 0.804 & 53.7 & 3218.0 \\ 68.4 & 0.71 & 34.1 & 3115.0 \\ 72.9 & 0.772 & 42.2 & 2970.0 \\ 66.2 & 0.572 & 35.7 & 2318.0 \\ 72.2 & 0.688 & 26.4 & 2259.0 \\ 68.7 & 0.783 & 32.9 & 2510.0 \\ 71.7 & 0.806 & 68.2 & 3354.0 \\ 71.6 & 0.751 & 64.2 & 2501.0 \\ 79.2 & 0.937 & 81.3 & 3053.0 \\ 82.7 & 0.96 & 66.3 & 2739.0 \end{pmatrix}$$


0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
70 75 80
80
70
60
50
40
30
70 75 80
3200
3000
2800
2600
2400
70 75 80
80
70
60
50
40
30
0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95
3200
3000
2800
2600
2400
0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95
3200
3000
2800
2600
2400
30 40 50 60 70 80


0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
70 75 80
80
70
60
50
40
30
70 75 80
3200
3000
2800
2600
2400
70 75 80
80
70
60
50
40
30
0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95
3200
3000
2800
2600
2400
0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95
3200
3000
2800
2600
2400
30 40 50 60 70 80