SDJ-Matsumoto


```
A = random_matrix(QQ, 4); A
```

```
[   0    0   -2    0]
[  -1    1    0    0]
[  -1  1/2    0  1/2]
[   1    0   -2    0]
```

```
A.rank()
```

```
4
```

```
B = A^(-1); B
```

```
[  -1    0    0    1]
[  -1    1    0    1]
[-1/2    0    0    0]
[  -1   -1    2    1]
```

```
show(B)
```

$$\left(\begin{array}{rrrr} -1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -\frac{1}{2} & 0 & 0 & 0 \\ -1 & -1 & 2 & 1 \end{array}\right)$$

```
latex(B)
```

```
\left(\begin{array}{rrrr}
-1 & 0 & 0 & 1 \\
-1 & 1 & 0 & 1 \\
-\frac{1}{2} & 0 & 0 & 0 \\
-1 & -1 & 2 & 1
\end{array}\right)
```

```
f=x^4-2*x+1
solve(f==0,x)
```

```
[x == -1/2*(I*sqrt(3) + 1)*(1/9*sqrt(3)*sqrt(11) + 17/27)^(1/3) +
1/9*(-I*sqrt(3) + 1)/(1/9*sqrt(3)*sqrt(11) + 17/27)^(1/3) - 1/3, x ==
-1/2*(-I*sqrt(3) + 1)*(1/9*sqrt(3)*sqrt(11) + 17/27)^(1/3) -
1/9*(-I*sqrt(3) - 1)/(1/9*sqrt(3)*sqrt(11) + 17/27)^(1/3) - 1/3, x ==
(1/9*sqrt(3)*sqrt(11) + 17/27)^(1/3) - 2/9/(1/9*sqrt(3)*sqrt(11) +
17/27)^(1/3) - 1/3, x == 1]
```

```
show(solve(f==0,x))
```

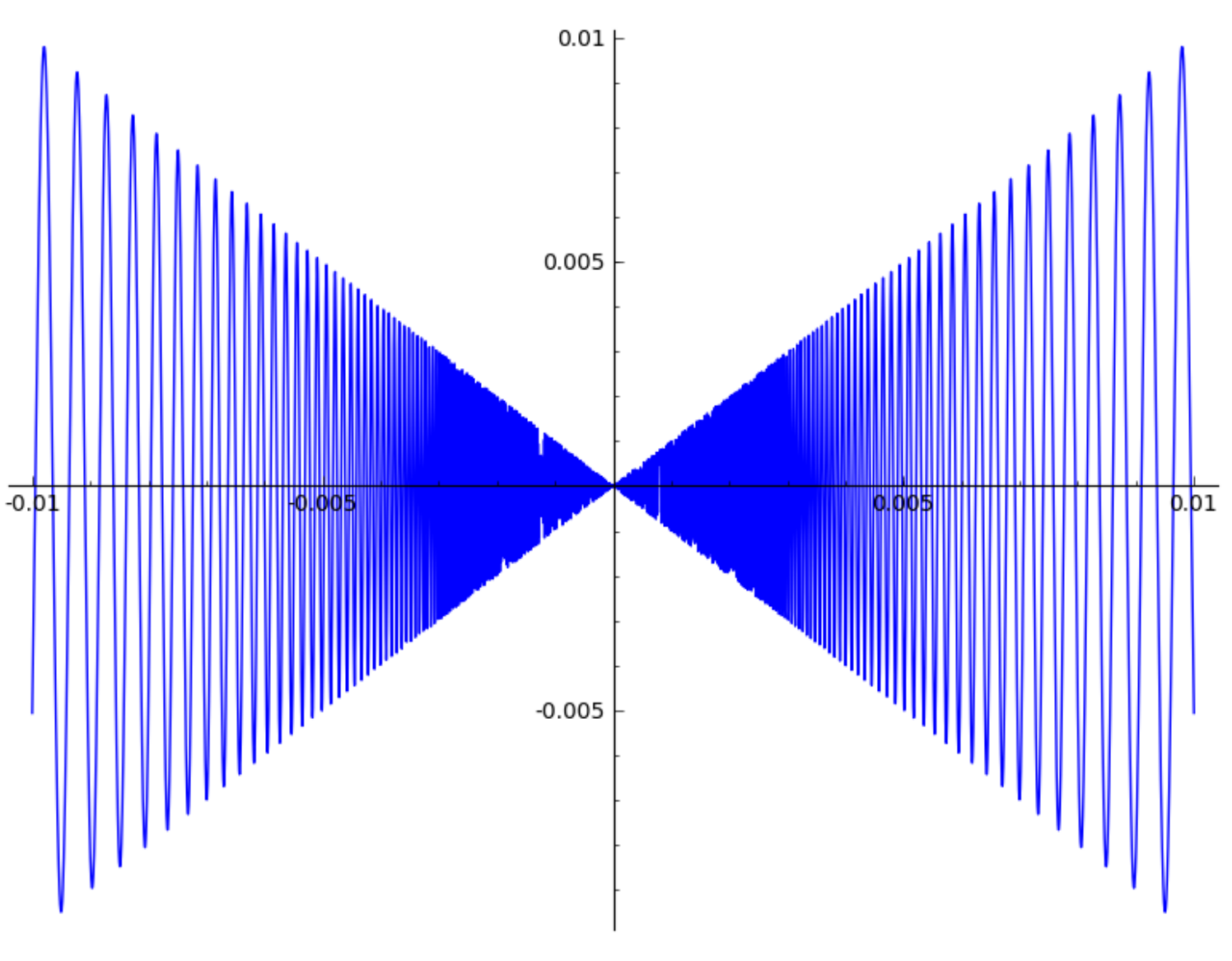
$$\left[x = -\frac{1}{2}\left(i\sqrt{3}+1\right)\left(\frac{1}{9}\sqrt{3}\sqrt{11}+\frac{17}{27}\right)^{\frac{1}{3}} + \frac{-i\sqrt{3}+1}{9\left(\frac{1}{9}\sqrt{3}\sqrt{11}+\frac{17}{27}\right)^{\frac{1}{3}}} - \frac{1}{3}, x = -\frac{1}{2}\left(-i\sqrt{3}+1\right)\left(\frac{1}{9}\sqrt{3}\sqrt{11}+\frac{17}{27}\right.\right.$$

```
f.integrate(x)
```

```
1/5*x^5 - x^2 + x
```

```
g=x*sin(1/x)
show(g)
plot(g,(x,-0.01,0.01))
```

$$x\sin\left(\frac{1}{x}\right)$$



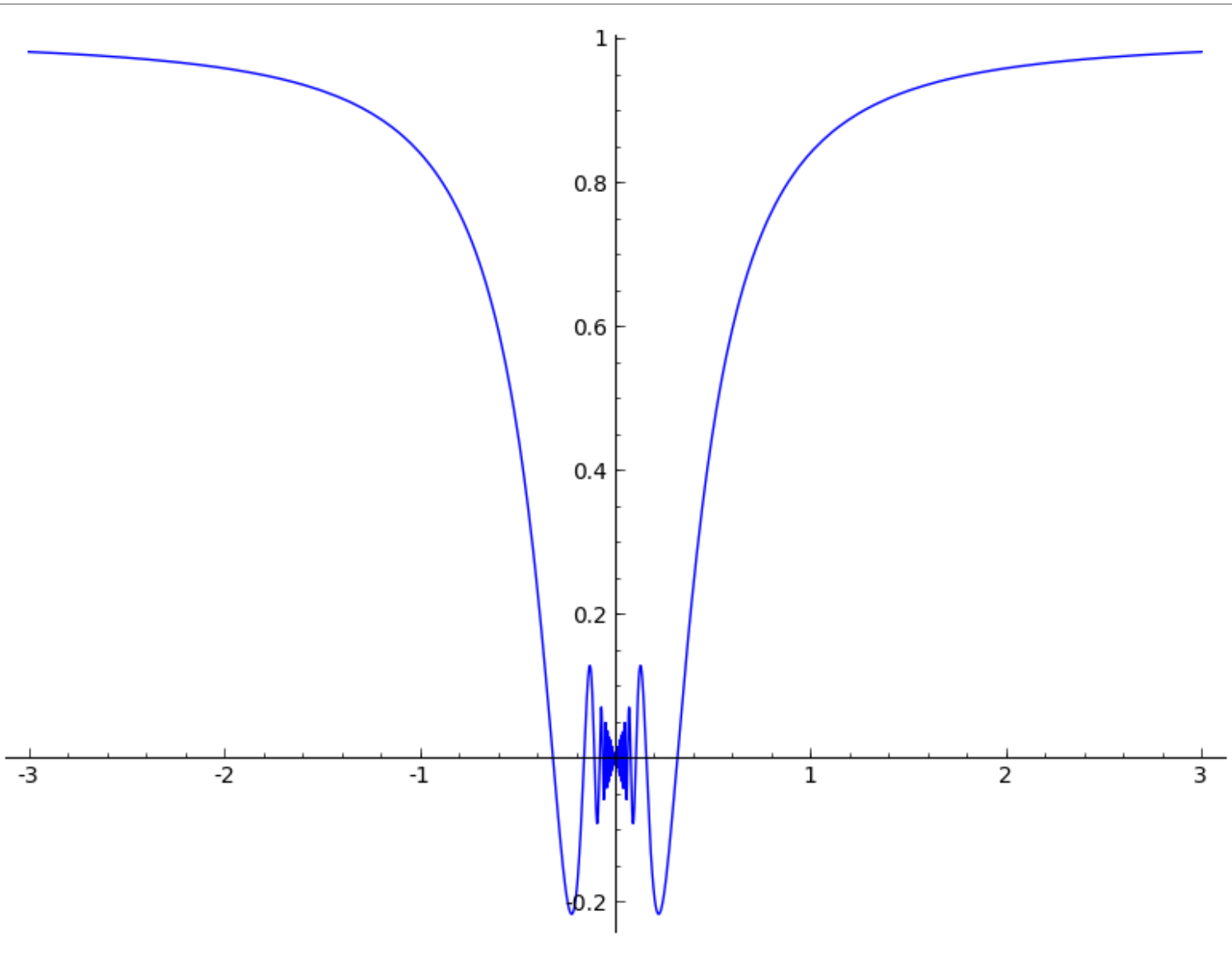

```
limit(g,x=0)
```

```
0
```

```
limit(g,x=-oo)
```

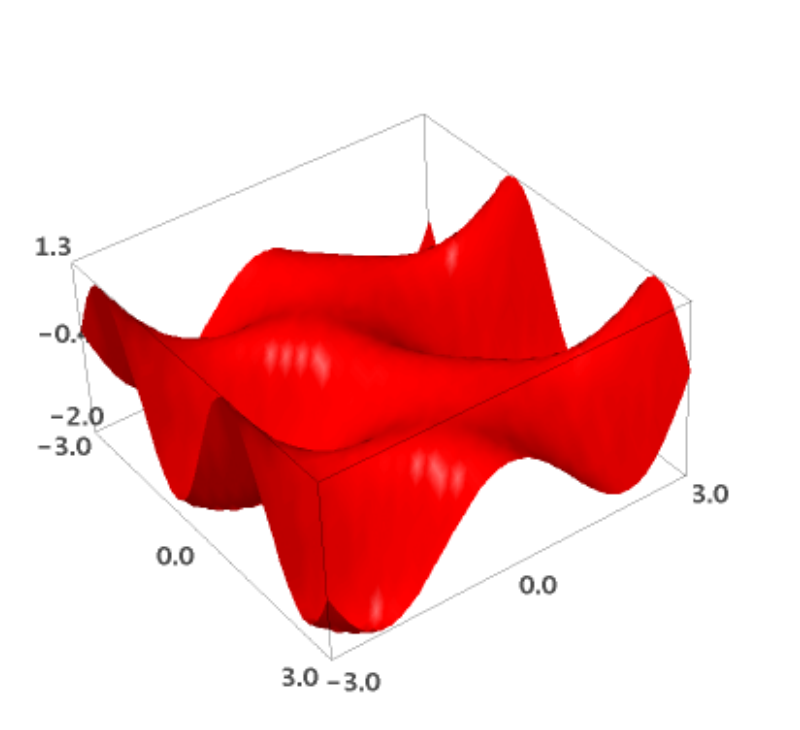
```
1
```

```
plot(g,(x,-3,3))
```




```
f(x,y) = sin(x-y)*y*cos(x)
plot3d(f, (x,-3,3), (y,-3,3), opacity=.9, color='red', figsize=3)
```

Toggle Advanced Controls | Help for Jmol 3-D viewer




```
%gp
factor(x^100-1)
```

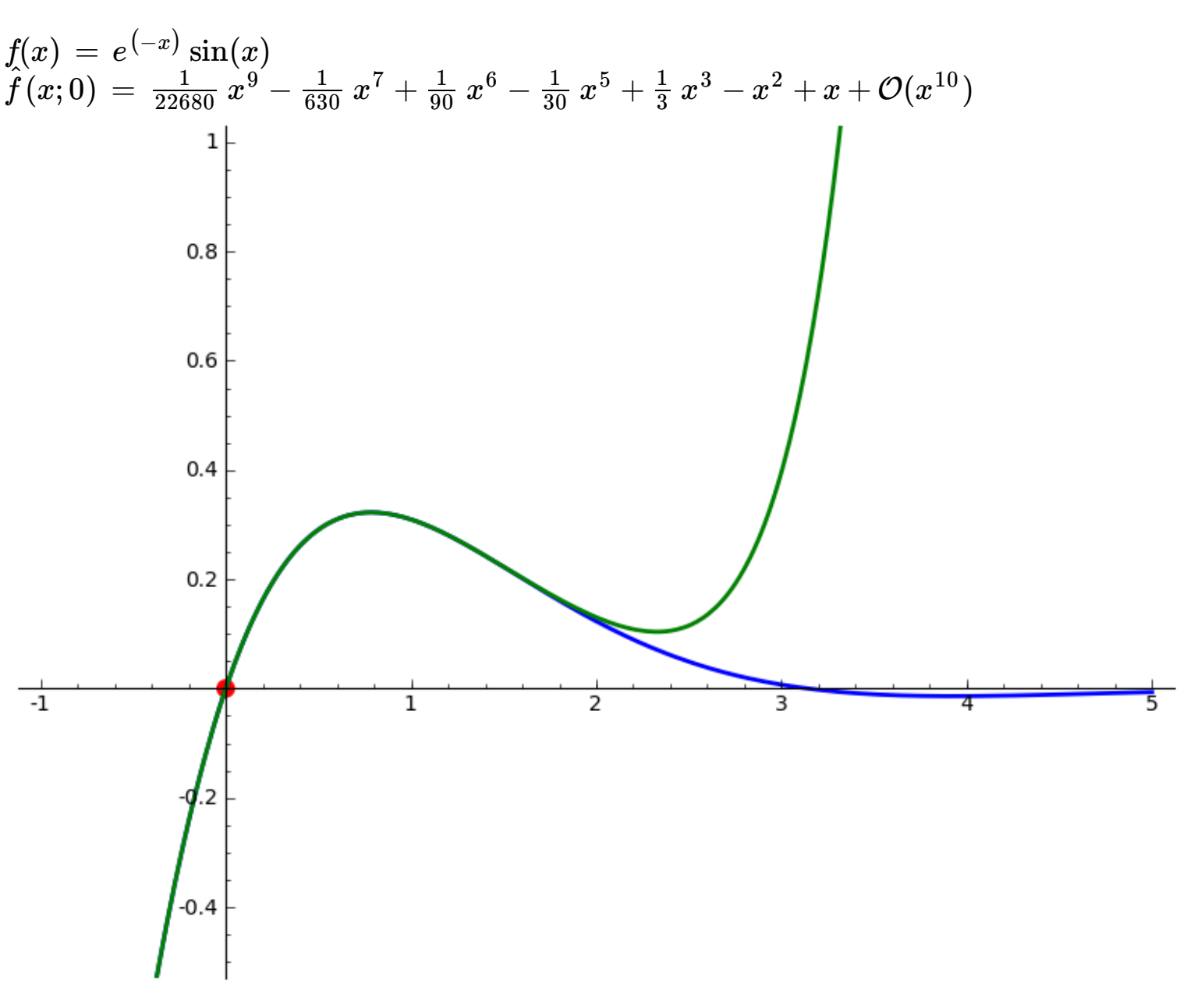
```
[x - 1 1]

[x + 1 1]

[x^2 + 1 1]

[x^4 - x^3 + x^2 - x + 1 1]

[x^4 + x^3 + x^2 + x + 1 1]

[x^8 - x^6 + x^4 - x^2 + 1 1]

[x^20 - x^15 + x^10 - x^5 + 1 1]

[x^20 + x^15 + x^10 + x^5 + 1 1]

[x^40 - x^30 + x^20 - x^10 + 1 1]
```

```
var('x') # Taylor-interact
x0  = 0
f   = sin(x)*e^(-x)
p   = plot(f,-1,5, thickness=2)
dot = point((x0,f(x=x0)),pointsize=80,rgbcolor=(1,0,0))
@interact
def _(order=(1..30)):
  ft = f.taylor(x,x0,order)
  pt = plot(ft,-1, 5, color='green', thickness=2)
  html('$f(x)\;=\;%s$'%latex(f))
  html('$\hat{f}(x;%s)\;=\;%s+\mathcal{O}(x^{%s})$'%(x0,latex(ft),order+1))
  show(dot + p + pt, ymin = -.5, ymax = 1)
```

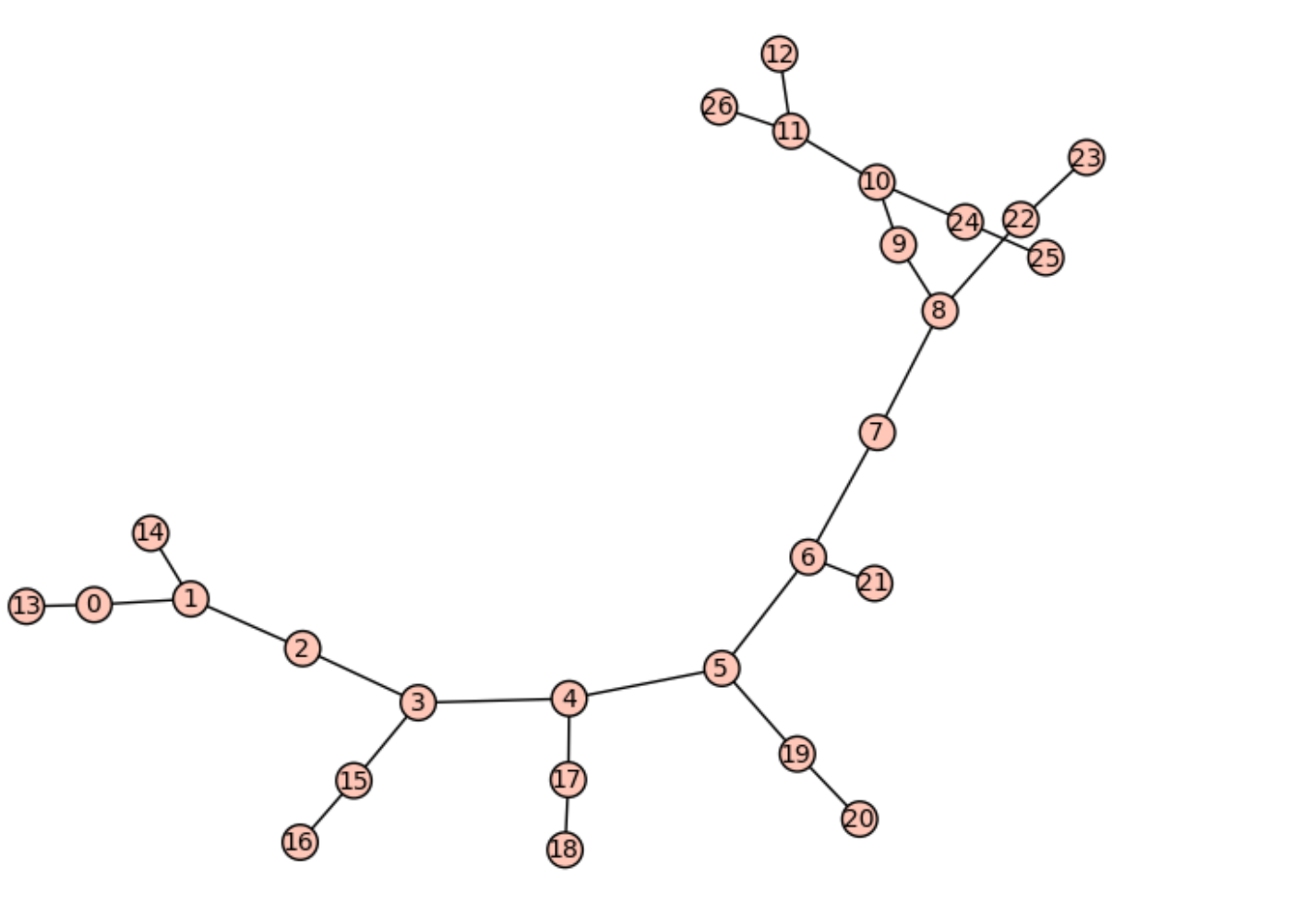
order 9

$f(x) = e^{(-x)}\sin(x)$

$\hat{f}(x;0) = \frac{1}{22680}x^9 - \frac{1}{630}x^7 + \frac{1}{90}x^6 - \frac{1}{30}x^5 + \frac{1}{3}x^3 - x^2 + x + \mathcal{O}(x^{10})$



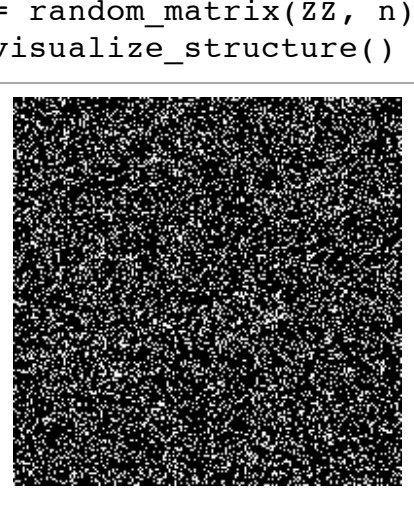

```
set_random_seed(1)
G = graphs.RandomLobster(8, .6, .3)
show(G, figsize=7)
```




```
G.shortest_path(13,20)
```

```
[13, 0, 1, 2, 3, 4, 5, 19, 20]
```

```
n = 200
C = random_matrix(ZZ, n)
C.visualize_structure()
```



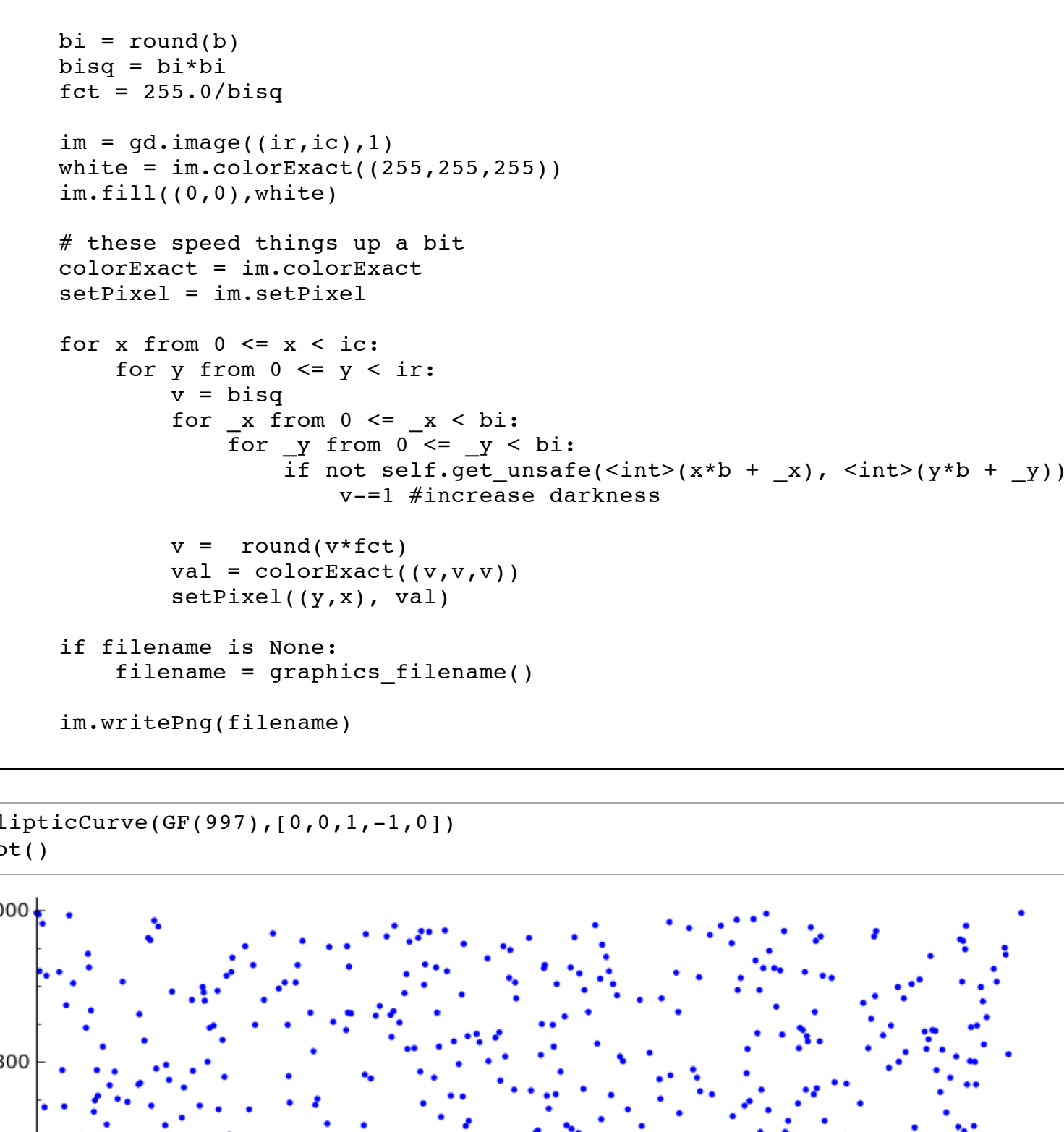
```
C.visualize_structure??
```

File: /Applications/sage/devel/sage/sage/matrix/matrix2.pyx

Source Code (starting at line 7349):

```
def visualize_structure(self, filename=None, maxsize=512):
    """
    Write a PNG image to 'filename' which visualizes self by putting
    black pixels in those positions which have nonzero entries.

    White pixels are put at positions with zero entries. If 'maxsize'
    is given, then the maximal dimension in either x or y direction is
    set to 'maxsize' depending on which is bigger. If the image is
    scaled, the darkness of the pixel reflects how many of the
    represented entries are nonzero. So if e.g. one image pixel
    actually represents a 2x2 submatrix, the dot is darker the more of
    the four values are nonzero.

    INPUT:


    -  ``filename`` - either a path or None in which case a
       filename in the current directory is chosen automatically
       (default:None)


    maxsize - maximal dimension in either x or y direction of the
    resulting image. If None or a maxsize larger than
    max(self.nrows(),self.ncols()) is given the image will have the
    same pixelsize as the matrix dimensions (default: 512)

    EXAMPLE::

        sage: M = random_matrix(CC, 4)
        sage: M.visualize_structure(os.path.join(SAGE_TMP, "matrix.png"))
    """
    import gd
    import os

    cdef int x, y, _x, _y, v, bi, bisq
    cdef int ir,ic
    cdef float b, fct

    mr, mc = self.nrows(), self.ncols()

    if maxsize is None:

        ir = mc
        ic = mr
        b = 1.0

    elif max(mr,mc) > maxsize:

        maxsize = float(maxsize)
        ir = int(mc * maxsize/max(mr,mc))
        ic = int(mr * maxsize/max(mr,mc))
        b = max(mr,mc)/maxsize

    else:

        ir = mc
        ic = mr
        b = 1.0

    bi = round(b)
    bisq = bi*bi
    fct = 255.0/bisq

    im = gd.image((ir,ic),1)
    white = im.colorExact((255,255,255))
    im.fill((0,0),white)

    # these speed things up a bit
    colorExact = im.colorExact
    setPixel = im.setPixel

    for x from 0 <= x < ic:
        for y from 0 <= y < ir:
            v = bisq
            for _x from 0 <= _x < bi:
                for _y from 0 <= _y < bi:
                    if not self.get_unsafe(<int>(x*b + _x), <int>(y*b + _y)).is_zero():
                        v-=1 #increase darkness

            v =  round(v*fct)
            val = colorExact((v,v,v))
            setPixel((y,x), val)

    if filename is None:
        filename = graphics_filename()

    im.writePng(filename)
```

```
E2=EllipticCurve(GF(997),[0,0,1,-1,0])
E2.plot()
```



evaluate