# TP2

```
M=matrix([[2,3,4],[3,4,5],[4,5,6]]);M
```

```
[2 3 4]
[3 4 5]
[4 5 6]
```

```
N=matrix(3,3,[(i+j) for i in range(1,4) for j in range(1,4)]);N
```

```
[2 3 4]
[3 4 5]
[4 5 6]
```

```
M=matrix([[1/2,1/3,1/4],[1/3,1/4,1/5],[1/4,1/5,1/6]]);M
```

```
[1/2 1/3 1/4]
[1/3 1/4 1/5]
[1/4 1/5 1/6]
```

```
N=matrix(3,3,[1/(i+j) for i in range(1,4) for j in range(1,4)]);N
```

```
[1/2 1/3 1/4]
[1/3 1/4 1/5]
[1/4 1/5 1/6]
```

```
def cauchy(n):
    return matrix(n,n,[1/(i+j) for i in range(1,n+1) for j in range(1,n+1)])
```

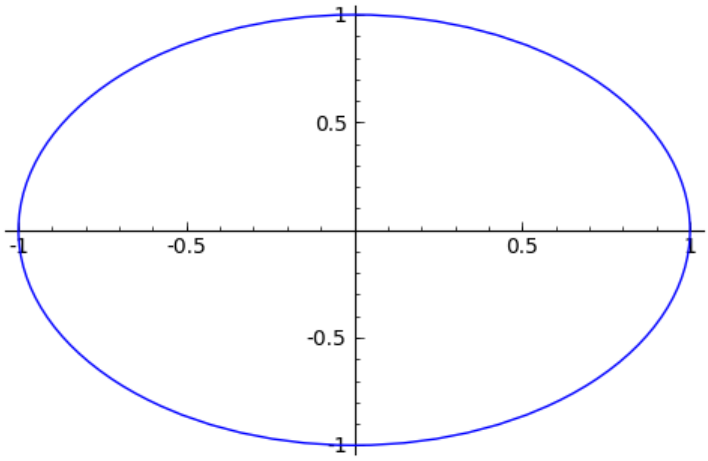
```
cauchy(3)
```

```
[1/2 1/3 1/4]
[1/3 1/4 1/5]
[1/4 1/5 1/6]
```

```
#Méthode de Monte-Carlo:
s=0
N=10^5
for i in range(N):
    x = random()
    y = random()
    if sqrt(x^2+y^2)<=1:
        s+=1
print n(s/N*4, digits=30)
```

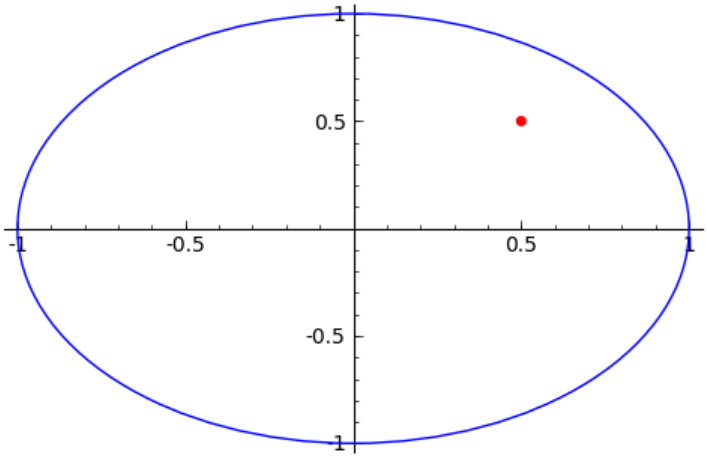
```
3.13680000000000000000000000000
```

```
c=circle((0,0), 1,color='blue')
c
```

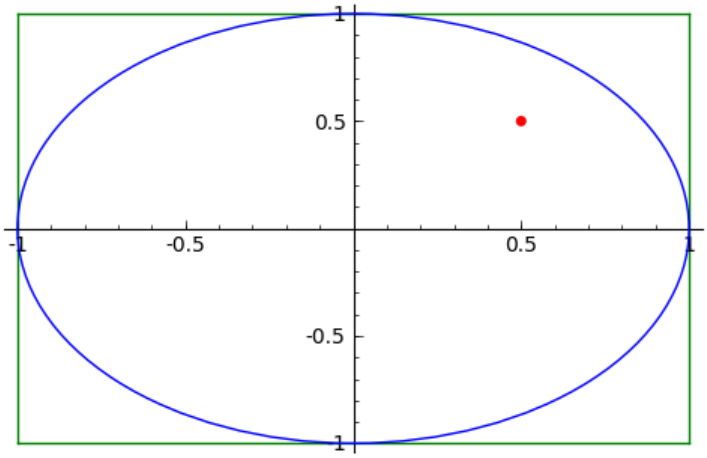



```
c += line([(1.0, 1.0),(1.0, -1.0)], color='green')
c += line([(1.0, -1.0),(-1.0, -1.0)], color='green')
c += line([(-1.0, -1.0),(-1.0, 1.0)], color='green')
```

```
c += line([(-1.0, 1.0),(1.0, 1.0)], color='green')
c
```




```
c += point((0.5, 0.5), color='red', pointsize=20)
c
```



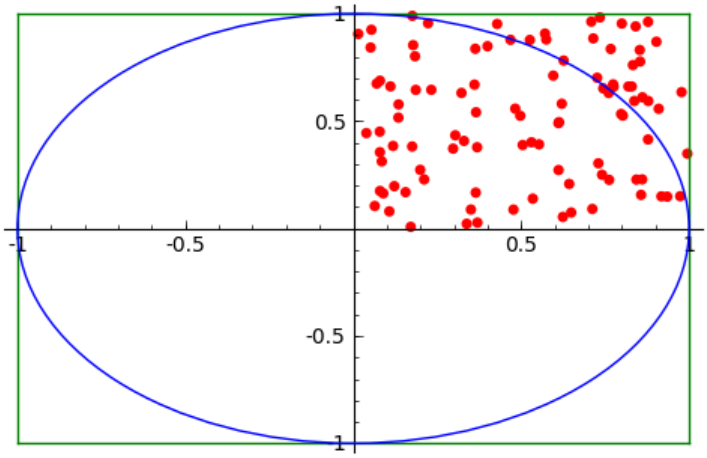

```
c=0;c
```

```
c=circle((0,0), 1,color='blue')
c += line([(1.0, 1.0),(1.0, -1.0)], color='green')
c += line([(1.0, -1.0),(-1.0, -1.0)], color='green')
c += line([(-1.0, -1.0),(-1.0, 1.0)], color='green')
c += line([(-1.0, 1.0),(1.0, 1.0)], color='green')
c += point((0.5, 0.5), color='red', pointsize=20)
c
```

```
#Méthode de Monte-Carlo:
s=0
N=100
c=0
c=circle((0,0), 1,color='blue')
c += line([(1.0, 1.0),(1.0, -1.0)], color='green')
c += line([(1.0, -1.0),(-1.0, -1.0)], color='green')
c += line([(-1.0, -1.0),(-1.0, 1.0)], color='green')
c += line([(-1.0, 1.0),(1.0, 1.0)], color='green')
for i in range(N):
    x=random()
    y=random()
    c += point((x, y), color='red', pointsize=20)
    if sqrt(x^2+y^2)<=1:
        s+=1
```

```
numerical_approx(pi, prec=35)
print "pi/4 =" , numerical_approx(pi/4, prec=35)
print "rapport entre le nb de pts tirés dans le disque et le nb de pts tirés au total
=", n(s/N,35)
c
```

```
pi/4 = 0.7853981634
rapport entre le nb de pts tirés dans le disque et le nb de pts
tirés au total = 0.7200000000
```




```
n(pi/4)
```

```
0.785398163397448
```

```
var('x')
g = cos(x)+x^2
n(g(2))
```

```
3.58385316345286
```

```
def bonjour(s):
    print s
    if s<=10:
        print('bonjour et bienvenue au cours')
    else:
        print('au revoir')

html("Ci-dessous un exemple de fonction interactive")
@interact
def _(p=(1,(0,120))):
    bonjour(p)
```

```
def point_fixe(g,x0,epsilon,N):
    n = 0
    p = x0
    q = g(x0).n()
    while ((abs(p-q)>epsilon) or (n<N)):
        q = g(p).n()
        p = q
        n += 1
    return(q)
```

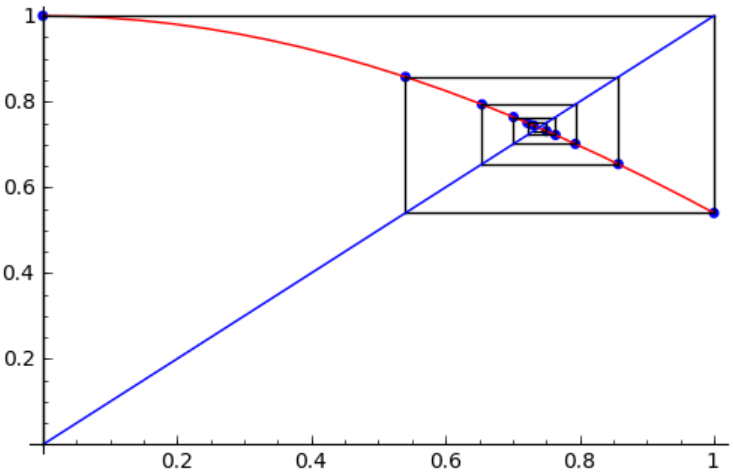
```
point_fixe(cos(x),0.0, 10^(-3),10)
```

```
0.731404042422510
```

```
def point_fixe(g,x0,epsilon,N,a,b):
    n = 0
    p = x0
    q = g(x0).n()
    A=0
    B=0
```

```
    A=plot(g,a,b,rgbcolor='red')
    A+=point((p,g(p)),color='blue', pointsize=20)
    A+=line([(p,g(p)), (p,0)],color='black')
    B=plot(x,a,b,rgbcolor='blue')
    while ((abs(p-q)>epsilon) or (n<N)):
        q = g(p).n()
        A+=point((q,g(q)),color='blue', pointsize=20)
        A+=line([(p,g(p)), (q,q)],color='black')
        A+=line([(q,g(q)), (q,q)],color='black')
        p = q
        n += 1
    show(A+B)
    return(q)
```

```
point_fixe(cos(x),0.0, 10^(-3),10,0.0,1.0)
```




```
0.731404042422510
```

```
html("<H1>Méthode du point fixe</H1>")
@interact
def _(g=sqrt((2*x)+3),x0=1.0,N=(3,(2..20)),a=0.0,b=4.0, eps=(-3,(-12,-1))):
    epsilon = 10^eps
    print(point_fixe(g,x0,epsilon,N,a,b))
```

```
def point_fixe(g,x0,epsilon,N,a,b):
    n = 0
    p = x0
    q = g(x0).n()
    A=0
    B=0
    A=plot(g,a,b,rgbcolor='red')
    A+=point((p,g(p)),color='blue', pointsize=20)
    A+=line([(p,g(p)), (p,0)],color='black')
    B=plot(x,a,b,rgbcolor='blue')
    while ((abs(p-q)>epsilon) or (n<N)):
        q = g(p).n()
        A+=point((q,g(q)),color='blue', pointsize=20)
        A+=line([(p,g(p)), (q,q)],color='black')
        A+=line([(q,g(q)), (q,q)],color='black')
        p = q
        n += 1
    show(A+B)
    return(q)
html("<H1>Méthode du point fixe</H1>")
@interact
def _(g=sqrt((2*x)+3),x0=1.0,N=(3,(2..20)),a=0.0,b=4.0, eps=(-3,(-12,-1))):
    epsilon = 10^eps
    print(point_fixe(g,x0,epsilon,N,a,b))
```

```
icosahedron()
```

Get Image

```
icosahedron?
```

**File:** /home/sage/sage/local/lib/python2.6/site-packages/sage/plot/plot3d/platonic.py

**Type:** <type 'function'>

**Definition:** icosahedron(center=(0, 0, 0), size=1, **kwds)

**Docstring:**

An icosahedron.

INPUT:

- `center` - (default: (0,0,0))
- `size` - (default: 1)
- `color` - a string that describes a color; this can also be a list of 3-tuples or strings length 6 or 3, in which case the faces (and oppositive faces) are colored.
- `opacity` - (default: 1) if less than 1 then is transparent

EXAMPLES:

```
sage: icosahedron()
```

Two icosahedrons at different positions of different sizes.

```
sage: icosahedron((-1/2,0,1), color='orange') + \
      icosahedron((2,0,1), size=1/2, aspect_ratio=[1,1,1])
```

```
icosahedron((-1/2,0,1), color='orange') + icosahedron((2,0,1), size=1/2, aspect_ratio=
[1,1,1])
```

Get Image