Condition number

In [1]:

```
#keep
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as pt
%matplotlib inline
```

Let's grab a 2×2 matrix A:

In [2]:

```
#keep
if 0:
    np.random.seed(17)
    A = np.random.randn(2, 2)
else:
    A = np.array([[3, 0], [0,1]], dtype=np.float64)

A
```

Out[2]:

```
array([[ 3.,  0.],
       [ 0.,  1.]])
```

And its inverse:

In [3]:

```
Ainv = la.inv(A)
Ainv
```

Out[3]:

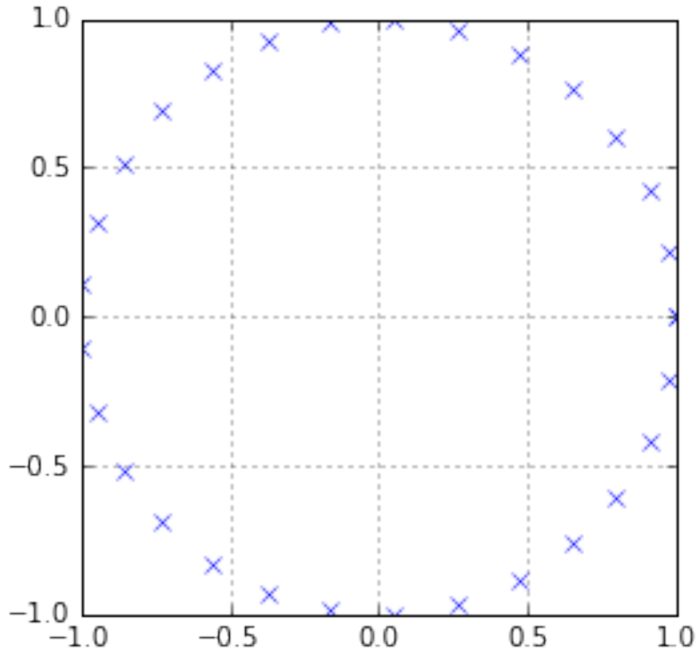
```
array([[ 0.33333333,  0.        ],
       [ 0.        ,  1.        ]])
```

Now we would like to figure out where that matrix puts all the vectors with 2-norm 1.

To do so, let's make an array of vectors with vectors with norm 1:

In [4]:

```
#keep
phi = np.linspace(0, 2*np.pi, 30)
xs = np.array([
    np.cos(phi),
    np.sin(phi)
])

pt.gca().set_aspect("equal")
pt.plot(xs[0], xs[1], "x")
pt.grid()
```



Now apply A to all those vectors...:

In [5]:

```
#keep
Axs = A.dot(xs)
Axs.shape
```

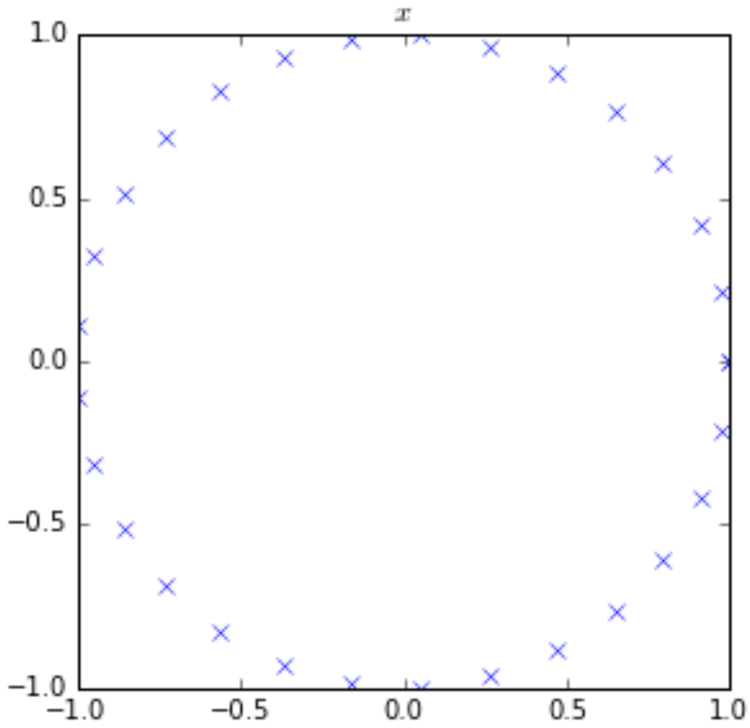
Out[5]:

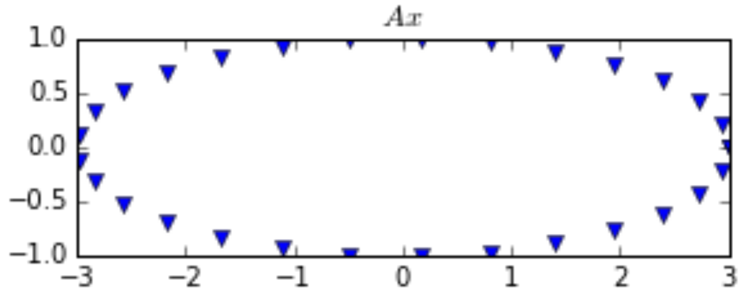
```
(2, 30)
```

...and plot:

In [12]:

```
#keep
pt.figure(figsize=(10, 5))

pt.subplot(121)
pt.title("$x$")
pt.plot(xs[0], xs[1], "x")
pt.gca().set_aspect("equal")

pt.subplot(122)
pt.title("$Ax$")
pt.plot(Axs[0], Axs[1], "v")
pt.gca().set_aspect("equal")
```





Next, let's see what happens to small perturbations at each of the x and Ax points.

To that end, let's make an array `ys` of shape $2 \times N_p \times N_p$, where N_p is the number of points above.

In [13]:

```
#keep
# ys has axes: XY x Npoints x Npoints

perturbation_size = 0.1
ys = perturbation_size * xs.reshape(2, -1, 1) + xs.reshape(2, 1, -1)

Ays = np.tensordot(A, ys, axes=1)
Ays.shape
```

Out[13]:

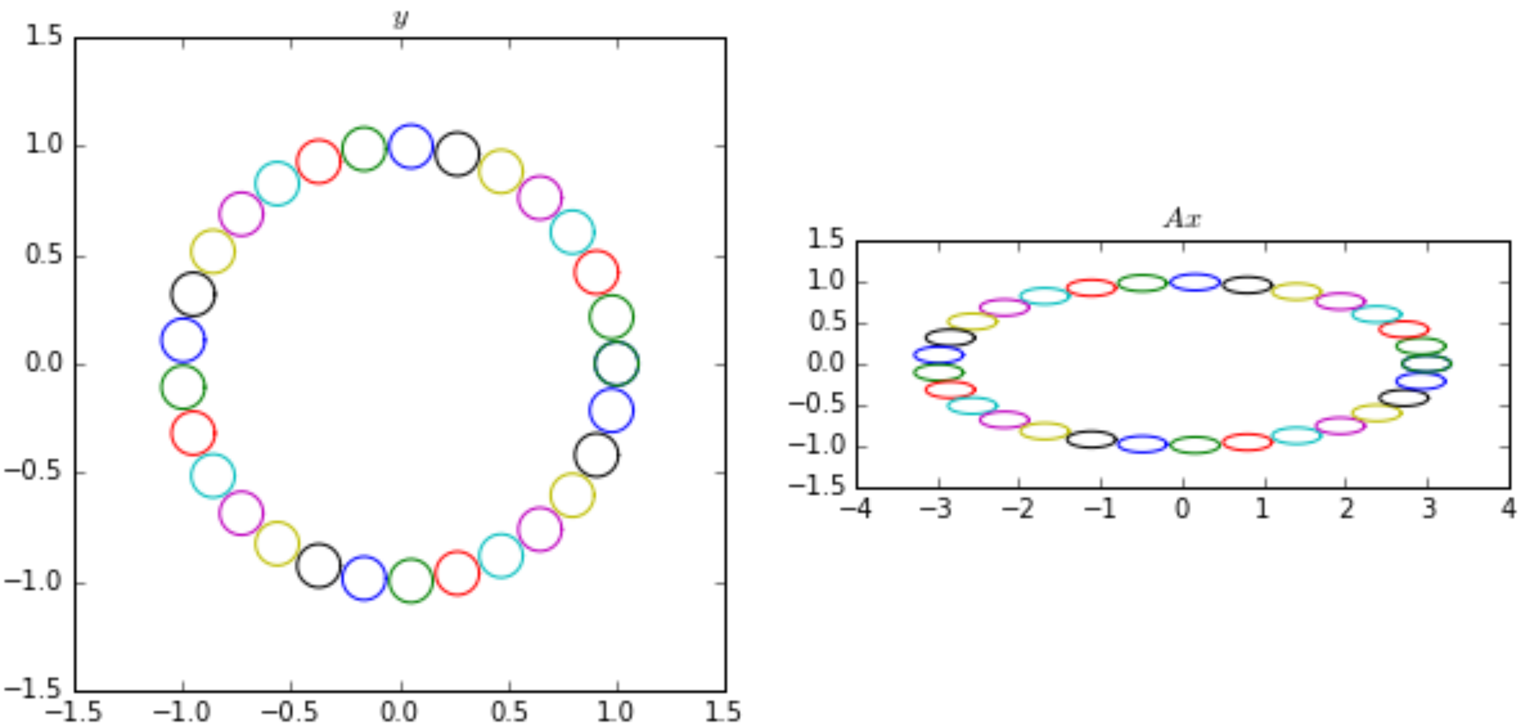
```
(2, 30, 30)
```

Side note: What does the argument `-1` to reshape do?

Let's plot what we've just made

In [14]:

```
#keep
pt.figure(figsize=(10, 5))

pt.subplot(121)
pt.title("$y$")
pt.plot(ys[0], ys[1])
pt.gca().set_aspect("equal")

pt.subplot(122)
pt.title("$Ax$")
pt.plot(Ays[0], Ays[1])
pt.gca().set_aspect("equal")
```



Let's compare this with $\|A\|$:

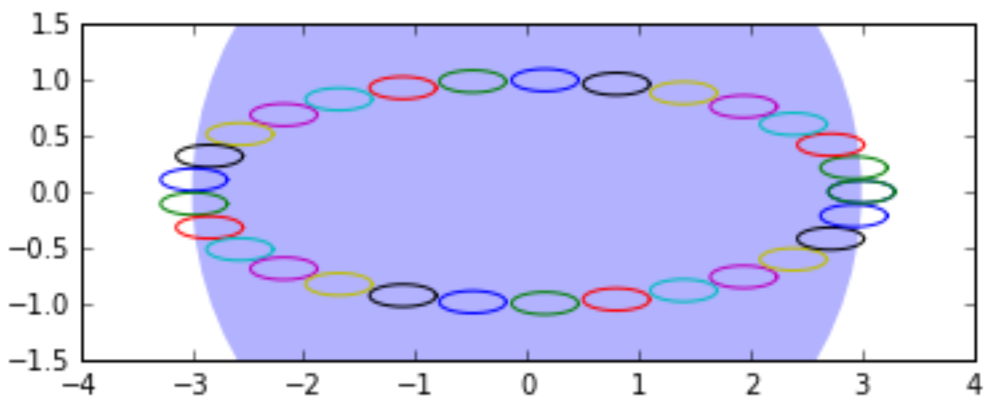
In [15]:

```
#keep
norm = la.norm(A, 2)
print(norm)

pt.plot(Ays[0], Ays[1])

ax = pt.gca()
ax.set_aspect("equal")
ax.add_artist(pt.Circle([0, 0], norm, alpha=0.3, lw=0))
```

3.0

Out[15]:

<matplotlib.patches.Circle at 0x10884cb90>



What we want now is a circle around each of the Ax that says,

"Because of the Δx variation, b is at most going to wiggle by this much, i.e. Δb will be at most this big."

Now we want a κ with $\frac{||\Delta b||}{||b||} \leq \kappa \frac{||\Delta x||}{||x||}$.

Assume $||x|| = 1$. Equivalent: $||\Delta b|| \leq \kappa ||\Delta x|| ||b||$.

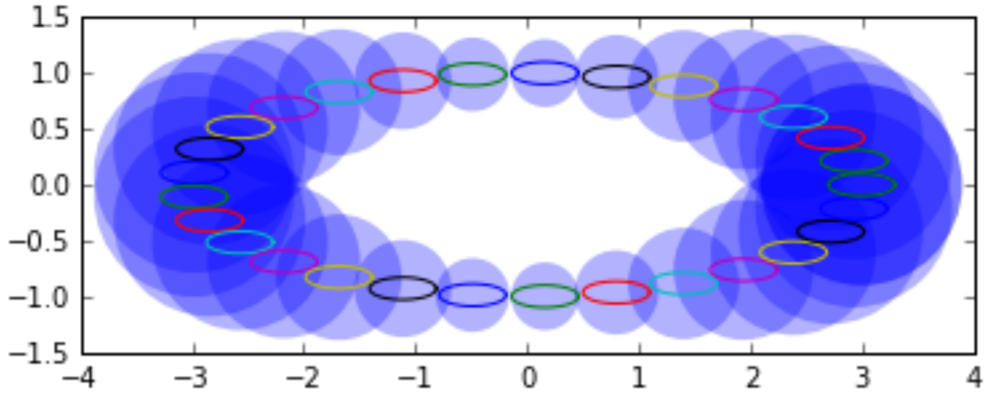
Which κ does the job?

In [16]:

```
kappa = la.norm(A, 2)*la.norm(Ainv, 2)
```

In [17]:

```
#keep
pt.plot(Ays[0], Ays[1])

ax = pt.gca()
ax.set_aspect("equal")
for i in range(Ays.shape[2]):
    b = Axs[:, i]
    norm_delta_y = kappa * perturbation_size * la.norm(b)
    ax.add_artist(pt.Circle(b, norm_delta_y, alpha=0.3, lw=0))
```



In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```