In [1]:

```
import numpy as np
import scipy.sparse as sparse
```

Let's make a *random* sparse matrix

First we'll set the density so that

$$density = \frac{nnz(A)}{n^2}$$

In [103]:

```
n = 1000
density = 5.0 / n # 5 points per row
nnz = int(n*n*density)
print(nnz)
```

```
5000
```

Now make the entries:

In [104]:

```
row = np.random.random_integers(low=0, high=n-1, size=nnz)
col = np.random.random_integers(low=0, high=n-1, size=nnz)
data = np.ones(nnz, dtype=float)

A = sparse.coo_matrix((data, (row, col)), shape=(n, n))
print(A.dtype)
```

```
float64
```

But let's make it positive definite:

In [105]:

```
A.data[:] = -1.0                    # -1 for off-diagonals
rowsum = -np.array(A.sum(axis=1))  # positive rowsum
rowsum = rowsum.ravel()
A.setdiag(rowsum)
```

In [106]:

```
u = np.random.rand(n)
v = np.random.rand(n)
```

In [107]:

```
%timeit v = A * u
```

```
The slowest run took 6.91 times longer than the fastest. This could
mean that an intermediate result is being cached
100000 loops, best of 3: 14.8 µs per loop
```

In [108]:

```
B = A.todense()
```

In [109]:

```
%timeit v = B.dot(u)
```

```
1000 loops, best of 3: 518 µs per loop
```

In [ ]:

```

```