Relative cost of matrix operations

In [1]:

```
#keep
import numpy as np
import scipy.linalg as spla
import scipy as sp

import matplotlib.pyplot as pt
%matplotlib inline

from time import time

np.alterdot()
```

In [2]:

```
#keep
n_values = (10**np.linspace(2, 3.25, 15)).astype(np.int32)
n_values
```

Out[2]:

```
array([ 100,  122,  150,  185,  227,  279,  343,  421,  517,  636,
781,
        959, 1178, 1447, 1778], dtype=int32)
```

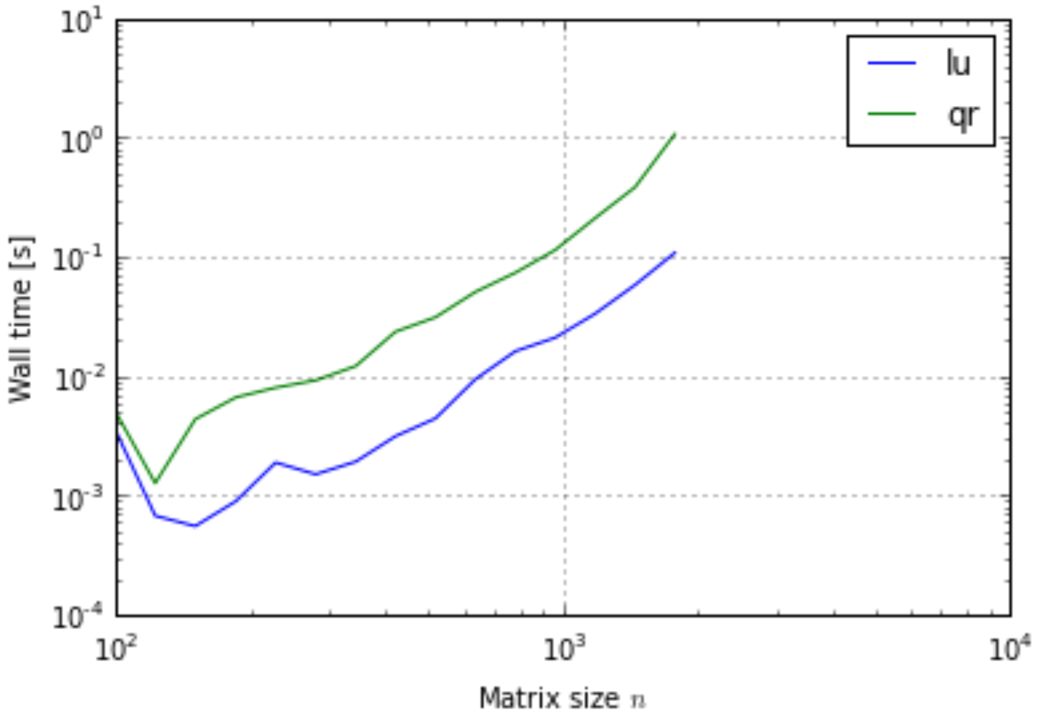
In [3]:

```
#keep

def mat_mul(A):
    return A.dot(A)

for name, f in [
        ("lu", spla.lu_factor),
        ("qr", spla.qr),
        ]:

    times = []
    print("----->", name)

    for n in n_values:
        print(n)

        A = np.random.randn(n, n)

        start_time = time()
        f(A)
        times.append(time() - start_time)

    pt.loglog(n_values, times, label=name)

pt.grid()
pt.legend(loc="best")
pt.xlabel("Matrix size $n$")
pt.ylabel("Wall time [s]")
```

```
-----> lu
100
122
150
185
227
279
343
421
517
636
781
959
1178
1447
1778
-----> qr
100
122
150
185
227
279
343
421
517
636
781
959
1178
1447
1778
```

```
Out[3]:
```

```
<matplotlib.text.Text at 0x111afb0f0>
```



- Can we see the asymptotic cost ($O(n^3)$) of these algorithms from the plot?

In [3]: