L10_notebook

September 21, 2022

[ ]:
```
import numpy as np
```

[ ]:
```
def house(A_param):
    A = A_param.copy()
    m, n = A.shape
    for k in range(n):
        x = A[k:, k]
        v = np.sign(x[0])*np.linalg.norm(x)*np.eye(1, len(x),0)+x
        v = v/np.linalg.norm(v)
        A[k:,k:] = A[k:,k:] - 2*np.dot(v.T, np.dot(v, A[k:,k:]))
    return A
```

[ ]:
```
# Random 5x5 matrix
A = np.random.rand(5,5)
```

[ ]:
```
np.set_printoptions(formatter={'float': lambda x: "{0:0.3f}".format(x)})
```

[ ]:
```
print(A)
```

```
[[0.763 0.909 0.163 0.771 0.737]
 [0.270 0.532 0.900 0.305 0.465]
 [0.021 0.615 0.117 0.564 0.321]
 [0.964 0.023 0.556 0.863 0.892]
 [0.513 0.920 0.752 0.782 0.204]]
```

[ ]:
```
print(house(A))
```

```
[[-1.359 -0.990 -0.951 -1.409 -1.221]
 [0.000 -1.164 -0.399 -0.477 -0.099]
 [0.000 0.000 0.814 0.006 0.033]
 [0.000 -0.000 0.000 -0.385 -0.137]
 [0.000 0.000 0.000 0.000 0.424]]
```

[ ]:

4) a) J : clockwise rotation

F : Reflects accross the $\theta/2$ line, measuring clockwise from the positive y-axis

b) Note - lots of credit to several online sets of notes on QR w/ givens rotations.

We need c, s such that

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}^T \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

With $r = \sqrt{a^2+b^2}$

Then

$$c = \frac{a}{\sqrt{a^2+B^2}} \qquad s = \frac{b}{\sqrt{a+2+b^2}}$$

Then for each i,j you want

to zero out you use the matrix G, which is the identity matrix but with: $G_{ii} = G_{jj} = c$, $G_{ij} = -G_{ji} = s$ and $i = j-1$, so that you always operate on adjacent rows, giving G a 2x2 sub-matrix differing from I.

Then for each element to be eliminated you apply:

$$A = G(c,s,i,j)^T A$$

which can be computationally simplified.

c) Because we know the structure of G, we can simplify the $G^T A$ calculation to only the small matrix calculation

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}^T \begin{bmatrix} a \\ b \end{bmatrix}$$

for 2 rows (supplying a & b) of each column of A. The other rows are unchanged.

This calculation has 4 multiplications and 2 additions per element, yielding 6ℓ flops where ℓ is the length of the row vector.

This is in comparison to 4ℓ via Householder Transformations.