SORBONNE UNIVERSITÉ

MODEL

QR decomposition and Hessenberg matrices

Méline TROCHON
Anatole VERCELLONI

Teacher : Jérémy Berthomieu

Novembre 2022 — Decembre 2022

Table des matières

1 questions :

Let consider $A \in \mathbb{C}^{n\times n}$ with λ_n, $n = 1, ..., n$ the eigenvalues of A and $Q \in \mathbb{C}^{n\times n}$ an orthogonal matrix, so $Q^*Q = Id$

So we have : $Av = \lambda_n v$
We will show that λ_n is an eigenvalue of QAQ^* :

$QAQ^*(Qv) = QA(QQ^*)v = QAv = \lambda Qv$
that is to say : $QAQ^*(Qv) = \lambda Qv$,
therefore λ_n is an eigenvalue with the eigenvector Qv
We are searching an rotation matrix G which vanish values in position (n, 1) of :
GA and GAG^* Let's take $G_{n-1,n} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & c & s \\ 0 & 0 & \cdots & -s & c \end{pmatrix}$
with $c = \frac{a_{n-1,1}}{r}$ and $s = \frac{a_{n,1}}{r}$ and $r = \sqrt{a_{n-1,1}^2 + a_{n,1}^2}$
We can compute AG and $A' = G_{n-1,n}AG_{n-1,n}^*$ and for both of them we got a zero in position (n,1)

as previously we can find $G_{n-2,n-1}$ = $\begin{pmatrix} 1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & c & s & 0 \\ 0 & \cdots & -s & c & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$
with $c = \frac{a_{n-2,1}}{r}$ and $s = \frac{a_{n-1,1}}{r}$ and $r = \sqrt{a_{n-2,1}^2 + a_{n-1,1}^2}$

to have a zero in (n,1) and in (n-1,1) for A' and $A'' = G_{n-2,n-1}A'G_{n-2,n-1}^*$

And to vanish the coefficient in position (n,2) we can take $G'_{n-1,n} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & c & s \\ 0 & 0 & \cdots & -s & c \end{pmatrix}$

with a difference for c and s : $c = \frac{a_{n-1,2}}{r}$ and $s = \frac{a_{n,2}}{r}$ and $r = \sqrt{a_{n-1,2}^2 + a_{n,2}^2}$

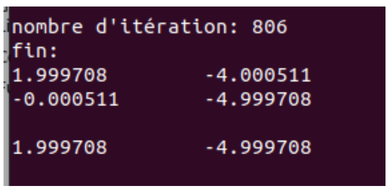
we can observe with this three question that with the application of some rotation matrices we are able to vanish some coefficient of A. And with keeping the correctness of the eigenvalues (question 1)

So we can think to an algorithm which is applying this transformation to make A upper-Hessenberg The difficulty is to respect the order of the transformations on A. So we choose to implement a recursive function, and it works well.

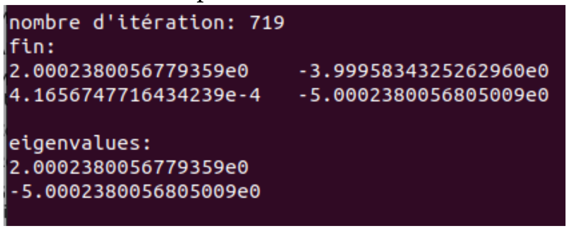
2 algorithms :

So we implements the different algorithm : Givens to get the QR factorization of a matrix, and the algorithm based on the sequence QAQ^* to get the eigenvalues of A, but also the algorithm based on the previous question to make a matrix upper_hessenberg

And we also implement the two first with mpfr to compare the accuracy of our result. So we test it on
$A = \begin{pmatrix} -5 & 0 \\ 4 & 2 \end{pmatrix}$
with 2 and -5 as eigenvalues and we get :// For the normal double floatting point precision :


```
nombre d'itération: 806
fin:
1.999708        -4.000511
-0.000511       -4.999708

1.999708        -4.999708
```

And for the mpfr one :



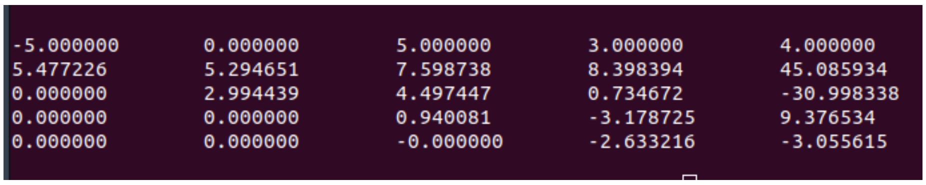
```
nombre d'itération: 719
fin:
2.0002380056779359e0    -3.9995834325262960e0
4.1656747716434239e-4   -5.0002380056805009e0

eigenvalues:
2.0002380056779359e0
-5.0002380056805009e0
```

So, as excepted, we can observe that the accuracy is better for mpfr but also the number of step

And we can have also test the algorithm which make A upper-Hessenberg with

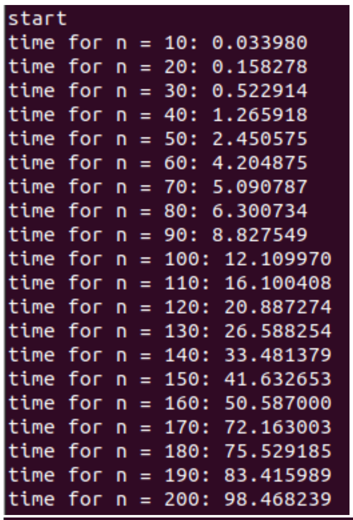
$$A = \begin{pmatrix} -5 & 0 & 5 & 3 & 4 \\ 4 & 2 & 3 & 5 & 55 \\ 2 & 2 & 2 & 6 & 7 \\ 1 & 2 & 3 & 4 & 0.94583 \\ 3 & 5 & 7.54 & 3.3333 & 4 \end{pmatrix}$$

and we got :

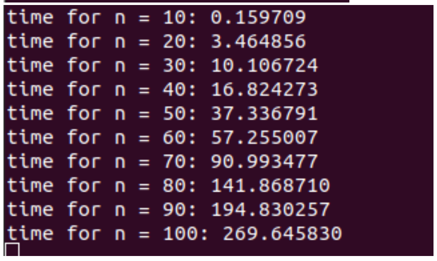


```
-5.000000       0.000000        5.000000        3.000000        4.000000
5.477226        5.294651        7.598738        8.398394        45.085934
0.000000        2.994439        4.497447        0.734672        -30.998338
0.000000        0.000000        0.940081        -3.178725       9.376534
0.000000        0.000000        -0.000000       -2.633216       -3.055615
```

And we also made some time test in double floating point number and with mpfr :



```
start
time for n = 10: 0.033980
time for n = 20: 0.158278
time for n = 30: 0.522914
time for n = 40: 1.265918
time for n = 50: 2.450575
time for n = 60: 4.204875
time for n = 70: 5.090787
time for n = 80: 6.300734
time for n = 90: 8.827549
time for n = 100: 12.109970
time for n = 110: 16.100408
time for n = 120: 20.887274
time for n = 130: 26.588254
time for n = 140: 33.481379
time for n = 150: 41.632653
time for n = 160: 50.587000
time for n = 170: 72.163003
time for n = 180: 75.529185
time for n = 190: 83.415989
time for n = 200: 98.468239
```



```
time for n = 10: 0.159709
time for n = 20: 3.464856
time for n = 30: 10.106724
time for n = 40: 16.824273
time for n = 50: 37.336791
time for n = 60: 57.255007
time for n = 70: 90.993477
time for n = 80: 141.868710
time for n = 90: 194.830257
time for n = 100: 269.645830
```

this is the time of the computation of the eigenvalues of a matrix with a size n.

So, we can observe that the time increase more in mpfr, it seems normal because mpfr is more accurate so it's normal that it takes more time.

3 Conclusion :

To conclude, we can say that we find a good way to compute the eigenvalues of a matrix with the QR-decomposition and the properties of orthogonal matrices. But also the possibility to make a matrix Hessenberg which could be very interesting because it is more benefit to make a matrix hessenberg then make it triangular (with QR) than to make it directly triangular. So these algorithms that we implements are workings, what we could improve may be the efficiency of some computation (matrix product, we could use strassen) if we want to apply that on larger matrices.