FYS3150 - Project 2

Vegard Falmår and Sigurd Sørlie Rustad
Universitetet i Oslo[*]
(Dated: 28. september 2020)

I. INTRODUCTION

II. THEORY

A. Unitary transformation

The transformed of a unitary matrix (U) is its inverse.

$$U^T = U^{-1}$$

From this we can prove that a unitary transformation preserves the orthonormality of vectors. Consider the set of orthonormal vectors $\{\mathbf{v}_i\}_i$ and the unitary transformation $\{U\mathbf{v}_i\}_i = \{\mathbf{w}_i\}_i$.

$$\begin{aligned} \mathbf{w}_i^T \mathbf{w}_j &= (U\mathbf{v}_i)^T U\mathbf{v}_j \\ &= \mathbf{v}_i^T U^T U \mathbf{v}_j = \mathbf{v}_i^T \mathbf{v}_j \\ &= \delta_{i,j} \end{aligned}$$

We notice that orthonormality is perserved.

B. Jacobi's rotation algorithm

Jacobi's rotation algorithm uses unitary transforms to diagonalize a matrix and preserves eigenvalues. A detailed description of the algorithm can be found here [1], however we will describe it briefly here. In order to diagonalize a given matrix A, as mentioned over, we perform a series of unitary transformations.

$$B = U_n^T U_{n-1}^T ... U_0^T A U_0 ... U_{n-1} U_n$$

Here U_i are the unitary matrices and B the resulting diagonal matrix. The geometric interpretation is that U_i performs a rotation on T in order to zero out elements. It turns out that the fastest way to do this, is to zero out the largest non-diagonal matrix-element. First We define:

$$\cot(\theta) = \tau = \frac{a_{ll} - a_{kk}}{2a_{kl}}.$$

Now to shorten notation we use $\tan(\theta) = t = s/c$, where $s = \sin(\theta) \wedge c = \cos(\theta)$. By defining θ such that a_{kl} becomes zero we get the quadratic equation

$$t^2 + 2\tau t - 1 = 0 \implies t = -\tau \pm \sqrt{1+\tau^2},$$

and can also obtain c and s

$$c = \frac{1}{1+t^2} \wedge s = tc.$$

The actual transformation is defined by the equations

$$\begin{aligned} b_{ik} &= a_{ik}c - a_{il}s, i \neq k, i \neq l \\ b_{il} &= a_{il}c + a_{ik}s, i \neq k, i \neq l \\ b_{kk} &= a_{kk}c^2 - 2a_{kl}cs + a_{ll}s^2 \\ b_{ll} &= a_{ll}c^2 + 2a_{kl}cs + a_{kk}s^2 \\ b_{kl} &= (a_{kk} - a_{ll})cs + a_{kl}(c^2 - s^2) \end{aligned}$$

Again see [1] for a more detail description of the algorithm.

III. METHOD

IV. RESULTS

V. DISCUSSION

VI. CONCLUSION

[*] sigurdsr@gmail.com; vegard@mail.no

[1] http://compphysics.github.io/ ComputationalPhysics/doc/pub/eigvalues/html/._eigvalues-bs011.html