Eigenvalues of updated matrix: Symmetric row/column update

Viktor Kristensen Haldborg
Student Nr: 201607139
AU ID: 540203

June 24, 2020

Introduction

I have been given the examination project (39 mod 22)=17 with the task of diagonalizing a matrix A given a matrix D and an update matrix W which gives rise to updated eigenvalues.

Theory

The matrix A to be diagonalized is given in the form

$$\mathbf{A} = \mathbf{D} + \mathbf{e}(p)\mathbf{u}^T + \mathbf{u}\mathbf{e}(p)^T \tag{1}$$

where $\mathbf{D}$ is a diagonal matrix with diagonal elements $\{d_i | i = 1, \ldots, n\}$, $\mathbf{e}(p)$ is the unit vector in the p -direction, and $\mathbf{u}$ is a given update vector where the p -th element can be assumed to equal zero, $u_p = 0$, without loss of generality. Expressing the eigenvalue equation in terms of equation (1) and subtracting $\lambda\mathbf{A}$ yields:

$$0 = (\mathbf{D} - \lambda)\mathbf{x} + \mathbf{e}(p)\mathbf{u}^T\mathbf{x} + \mathbf{u}\mathbf{e}(p)^T\mathbf{x} \tag{2}$$

We are ai priori given that $u_p = 0$, so for $k = p$ we get:

$$0 = (d_p - \lambda)\, x_p + \sum_{k=1}^{n} u_k x_k \tag{3}$$

and for $k \neq p$ we get:

$$0 = (d_k - \lambda)\, x_k + u_k x_p \tag{4}$$

By summation of all the elements from $k = 1$ to n, equation (7) can be expressed as:

$$0 = \sum_{k=1}^{n} u_k x_k + \sum_{k \neq p}^{n} \frac{u_k^2 x_p}{(d_k - \lambda)} \tag{5}$$

Substitution of equation (3) gives the secular equation written in the form:

$$0 = -(d_p - \lambda) + \sum_{k \neq p}^{n} \frac{u_k^2}{(d_k - \lambda)} = P_A \tag{6}$$

Solutions to this equation will yield the updated eigenvalues. The secular equation (6) is ill defined for $\lambda = d_k \forall k \neq p$.

The polynomial will not be differentiable on all $\mathbb{R}$, however it will be piece-wise differentiable in between the singularities:

$$1 + \sum_{k \neq p}^{n} \frac{u_k^2}{(d_k - \lambda)^2} \tag{7}$$

Which yields a strictly positive function defined on the same domain as equation (6).

By pulling out the n'th term in the sum of (6) and multiplying by $\lambda - d_n$;

$$0 = u_n^2 + (d_n - \lambda) \left[-(d_p - \lambda) + \sum_{k \neq p}^{n-1} \frac{u_k^2}{(d_k - \lambda)} \right] \tag{8}$$

it becomes evident, by the factoring of $(d_n - \lambda)$, that $\lambda = d_n$ will be a root of this polynomial if $u_n = 0$. However by default, a computer would zero out the term without any subsequent notion of its domain. To circumvent this it is clear that by multiplication;

$$\left(\prod_{k \neq p}^{n} (d_k - \lambda) \right) \left[-(d_p - \lambda) + \sum_{k \neq p}^{n} \frac{u_k^2}{(d_k - \lambda)} \right] \tag{9}$$

equation (9) may be constructed and used by the computer to have a continuously defined polynomial to work on whilst still retaining all information about the eigenvalues in (6).

The differentiated expression is found by application of the product rule:

$$\left(\prod_{k \neq p}^{n} (d_k - \lambda) \right) \left(1 + \sum_{k \neq p}^{n} \frac{u_k^2}{(d_k - \lambda)^2} - \left(\sum_{k \neq p}^{n} \frac{1}{(d_k - \lambda)} \right) P_A \right) \tag{10}$$

Numerical solutions

To obtain the roots to (6) the method of root finding using the Newton Raphson method is implemented and built for multidimensional purposes. This is done under the assumption that no degeneracy is introduced by the update. The function used is the one obtained in (6) and the multidimensional Newton Raphson method has been coded using the analytical derivative found from the secular equation (7). As such, an interval search can be brought to light since knowledge about the placement of the roots can be used to quickly generate an initial guess and break out of the given interval search when the root is obtained. When $u_k = 0$ for some k different from p the computer changes the function by zeroing out the $k'th$ term, and an addition to the code has thus been made to add in the missing eigenvalues which then must be exactly located outside the interval search, i.e on the given discontinuity d_k. If the trivial case is generated for which the update matrix is 0 all along the diagonal the main body of the code(interval search) will be skipped. A plethora of information for the specific implemenentations in A) and B) is given in the folders.

For the scenario of diagonalizing a matrix A for which the update gives rise to degeneracy in the eigenvalues, a different method has been brought to use. The reexpression of the secular equation obtained in (9) along with its derivative (10) has been implemented to search for the roots by "shooting" out initial guesses and then letting the Newton Raphson method converge to the roots. A sorting mechanism has then been implemented to sort the list of converged results. To find the degeneracy of a given root the function and its analytical derivative is used to find eigenvalues with degeneracy up to 3.

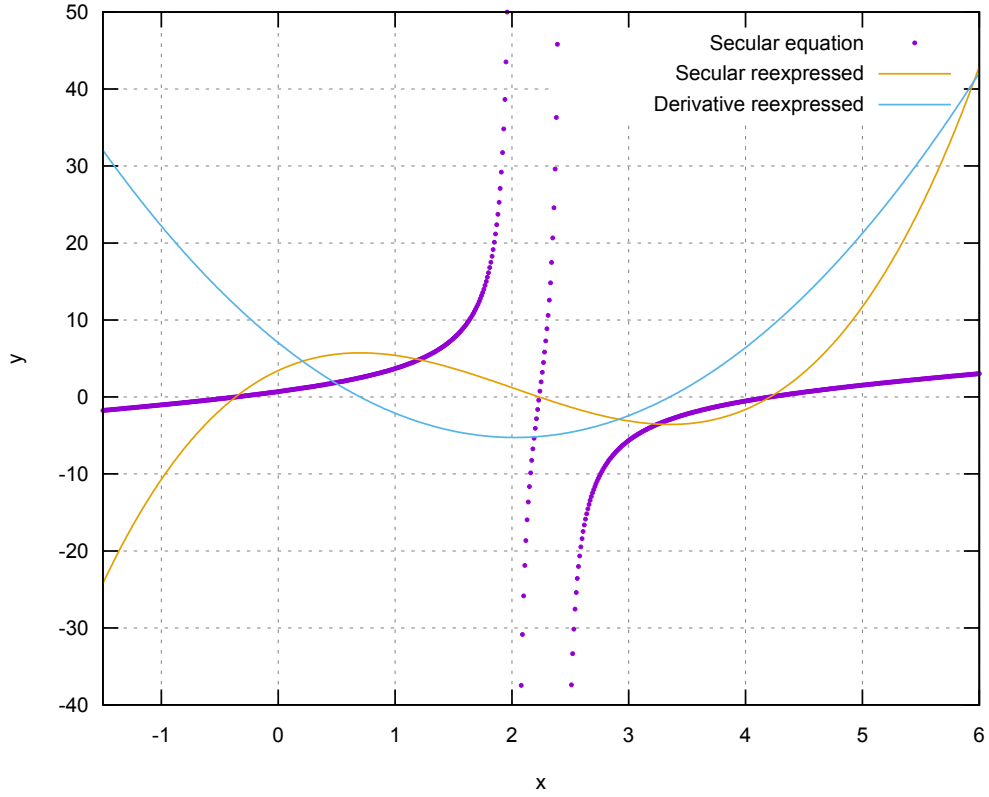

Figure 1: Plot of the secular equation obtained in (6), the reexpressed form (9) along with its derivative (10) for n=3, no degeneracy and n-1 discontinuities.

The results are presented in .txt files for the A),B) and C) part and are compared with the diagonalization of the same matrix using the Jacobi sweeps.

Discussion

Different opportunities were present, when trying to find the roots to the secular equation, such as bisection or simply brute forcing a solution by linearly running through the intervals and redefining the search based on if statements. However this would give rise to running through the listed elements more than once. Another way which could be achieved more easily was using binary

search $O(log(n))$ since the intervals would be ordered by default when searching the original secular equation. However the high resolution needed in the listing of points and the fact that the points would have to be run through and listed beforehand makes it redundant even as a guessing tool if no list is given beforehand.

Using the re-expressed secular equation to form a general basis to find solutions to the original secular equations seems to be the best way to go about treating the problem numerically given the ill defined points in the original secular equation. However given the made assumptions in part A) and B) the technique searching the intervals proved to have much greater stability when scaled, and much less tweaking of parameters was required since it would be fairly easy to find an adequate guess to be within convergence radius of the Newton's method.

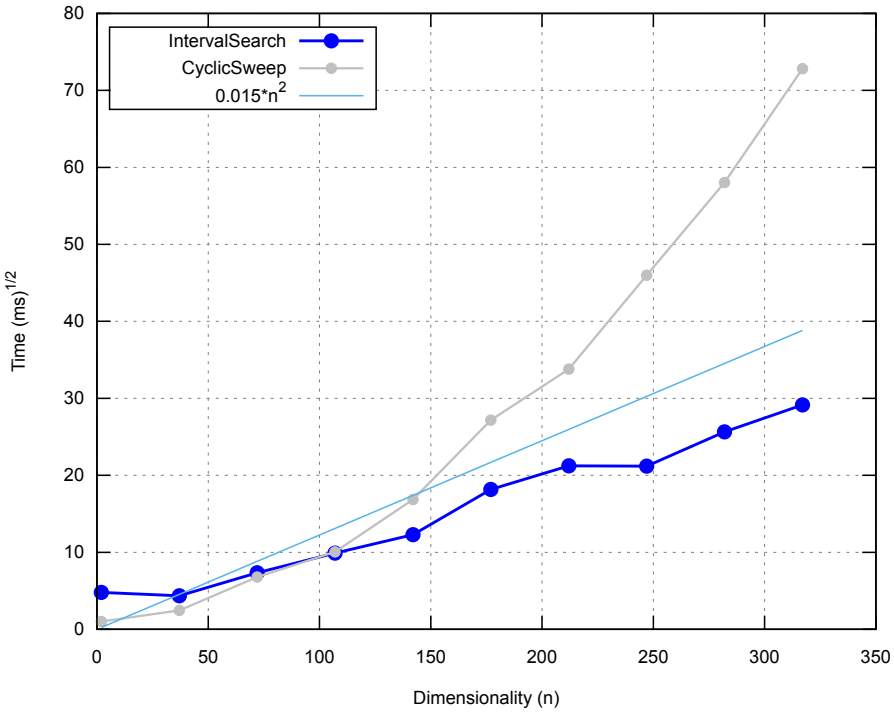

Figure 2: Plot of the operation time as a function of the dimensionality of the problem. It shows the operation time for the implemented interval search, for the assumptions made in part A), along with the diagonalization of the same matrices using Jacobian cyclic sweeps. Alongside it is a reference function $0.015 * n^2$ for which the interval search seems to lie well within. The cyclic Jacobian seems to be faster for low dimensions and then takes off showing its $O(n^3)$ dependancy.

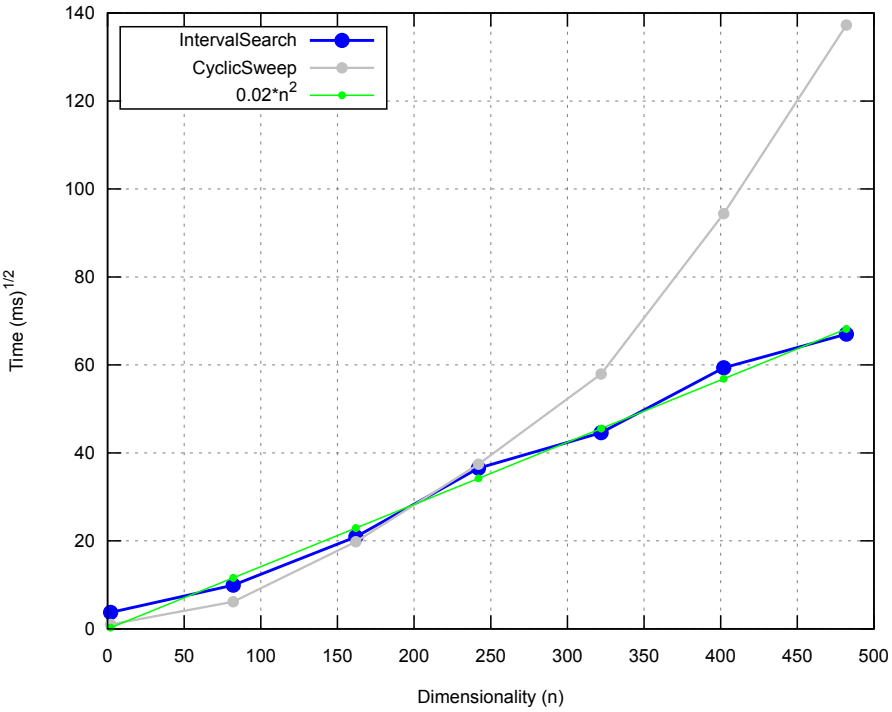

Figure 3: Plot of the operation time as a function of the dimensionality of the problem. It shows the operation time for the implemented interval search, for the assumptions made in part B), along with the diagonalization of the same matrices using Jacobian cyclic sweeps. Alongside it is a reference function $0.02 * n^2$ for which the interval search seems follow in a linear fashion, as expected. An additional reorganization of the u vector had to be done and therefore another search through n elements was needed, which explains the higher operation time and n dependance compared to the operation time displayed in figure (2).

References

[1] Dimitri.V. Fedorov, "Yet Another Introduction to Numerical Methods", 2020.