Implementation and analysis of the 'Divide and Conquer' method for the symmetric eigenvalue problem

Fran Špigel

Tuesday 12th February, 2019

1 Overview

Divide and Conquer (DAC for short) is a recursive eigenvalue algorithm applicable to symmetric, tridiagonal matrices. We can use it to decompose an arbitrary symmetric matrix by first applying a number of Householder reflections in order to reduce it to a tridiagonal form.
Let T be a tridiagonal, symmetric matrix such as

$$T = \left[\begin{array}{cccc|cccc} a_1 & b_1 & & & & & & \\ b_1 & \ddots & \ddots & & & & & \\ & \ddots & a_{m-1} & b_{m-1} & & & & \\ & & b_{m-1} & a_m & b_m & & & \\ \hline & & & b_m & a_{m+1} & b_{m+1} & & \\ & & & & b_{m+1} & \ddots & \ddots & \\ & & & & & \ddots & a_{n-1} & b_{n-1} \\ & & & & & & b_{n-1} & a_n \end{array}\right]$$

By transforming T into a block-diagonal form and a rank-1 correction, each block representing a smaller tridiagonal matrix, we have reduced the problem to two smaller problems, dimensions m and $n - m$, respectively. We then recursively apply the DAC algorithm to the sub-matrices in question, acquiring Q_1, Q_2, Λ_1 and Λ_2 such that the following is true:

$$T = \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix} \left(\begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix} + b_m u u^T \right) \begin{bmatrix} Q_1^T & \\ & Q^T \end{bmatrix} \tag{1}$$

Now all that remains is to update the eigenvalues Λ by solving the eigenvalue problem for the expression

$$\hat{D} = D + \rho u u^T,$$

where D is a diagonal matrix. This problem comes down to finding the roots of a secular equation, i.e. solving the equation

$$1 - \rho \sum_{i=1}^{n} \frac{u_i^2}{d_i - \lambda} = 0 \tag{2}$$

There are numerous numeric algorithm for finding roots, however the secular equation is tricky in that a number of vertical asymptotes appear along

the discontinuities d_i in its domain. It is for this reason, and certain other properties of the equation, that Newton's method is unsuitable for solving this problem. For the sake of simplicity, our implementation uses a simple binary search mechanism to find the roots in question. More on the unsuitability of Newton's method in section 3. A more detailed desciption of the algorithm may be found in N. Bosner's "PHD Lecture on Numerical Analysis" [1] and J. Demmel's "Applied numerical linear algebra" [2].

2 Implementation

We have implemented the method described above in MATLAB, by separating functionalities into different functions.

- *eig_by_DAC*
  - input:
    - A: an arbitrary, square, symmetrical matrix
    - tol - tolerance against which the solution should be measured
  - output:
    - Q - an orthonormal matrix such that each row is an eigenvector of A
    - L - a diagonal matrix such that $L_i i$ is the i-th eigenvalue of A (corresponding to the i-th row of Q. Q and L satisfy the equation $QLQ^* = A$
  - *eig_by_DAC* is the over-arching program for solving the eigenproblem of a general symmetrical matrix
- *DAC* - the core "Divide and Conquer" algorithm for solving the eigenproblem of a symmetrical tridiagonal matrix
- *secular_roots* - finds the roots of the secular equation encountered in the algorithm by performing a binary search in each area between two neighboring discontinuities d_i and d_{i+1}, where $i = 1, 2, ..., n-1$, assuming $d_i \neq d_{i+1}$ for each i

- *binary_root_search* - the binary root search described briefly below. The inputis a function f, two boundaries a and b such that the function is continuous on the interval $< a, b >$, and at least one root exists on that interval, and a tolerance tol. The output is a root x such that $|f(x)| < tol$. Note that f need not be contunuous on the points a and b themselves. The program begins by finding a_1 and b_1 such that f is continuous over $[a_1, b_1]$, and such that $f(a_1)f(b_1) \leq 0$, then looks for the root in that interval as per a standard binary search.

- *householder_generator* - generates a Householder reflector required to transform a symmetrical matrix into tridiagonal form

- *tridiagonalize* - transforms a matrix into tridiagonal form by applying a series of Householder reflectors

3 Solving the Secular Equation

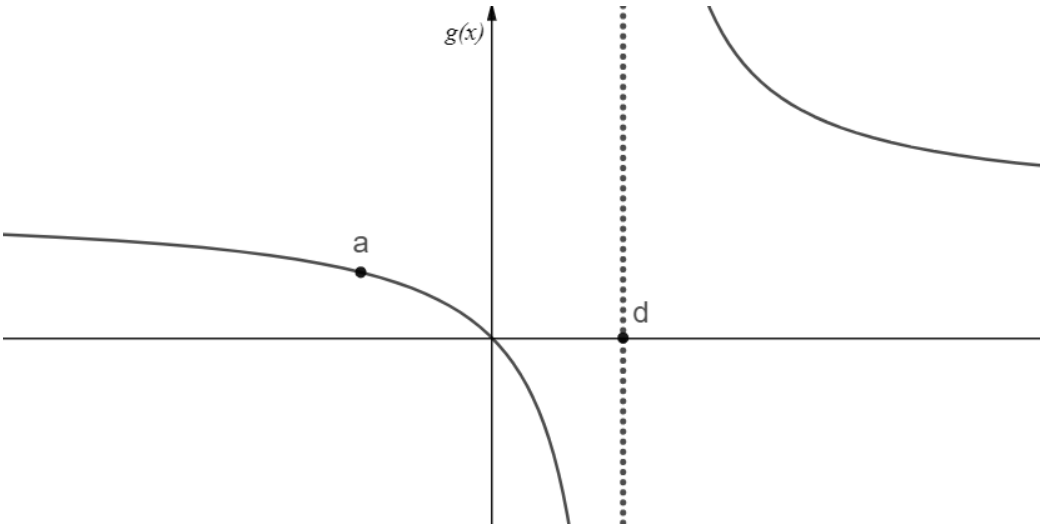
As mentioned previously, finding the roots of the secular function (2) is a tricky undertaking due to the prevalence of discontinuities in its domain, as well as the slant of the function between each pair of neighboring discontinuities.
To be specific, let $f : \mathbb{R} \to \mathbb{R}$ be defined as in (2), $\rho > 0$, and $d_1 < d_2 < ... < d_n$. It follows that f has a discontinuity at each point d_i, and n roots x_i such that $d_1 < x_1 < d_2 < x_2 < ... < d_n < x_{n-1}$.
The suggested method for acquiring these roots is Newton's method, which consists of relatively simple iterations of the form

$$x_{k+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3}$$

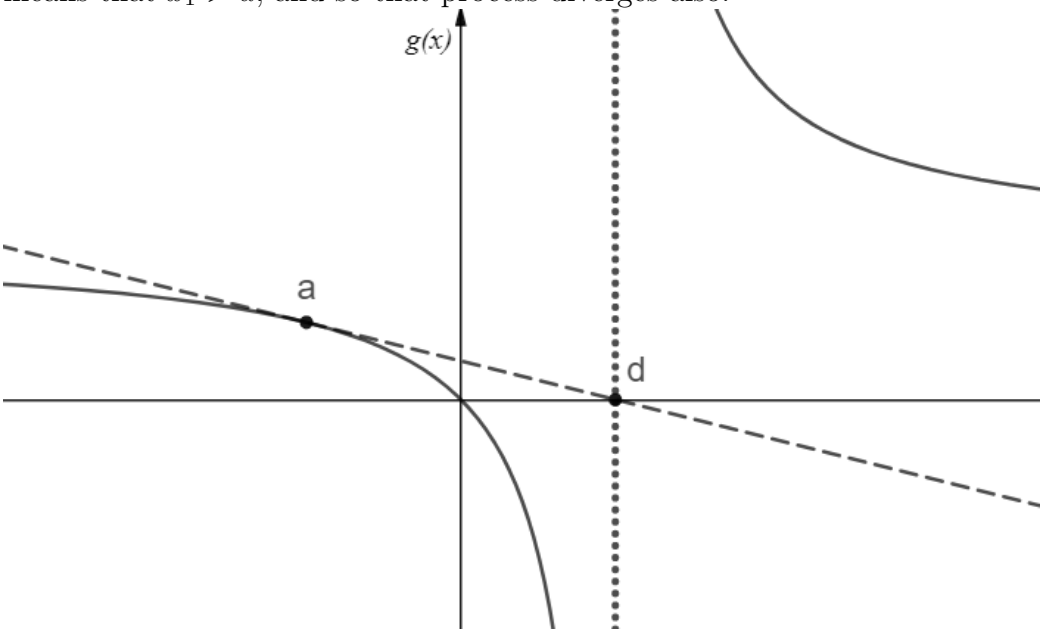
We will show here that there exists a large class of circumstances in which this approach is inadequate. First, let us observe a simpler secular function:

$$g(x) = 1 - \frac{u^2}{d - x}, \tag{4}$$

where $u, d \neq 0$, and its graph:



It is clear from the image that starting Newton's iterations at any point $x_0 > d$ will result in divergence towards positive infinity. It follows that any Newton process that eventually reaches that area will suffer the same fate. It can easily be calculated that starting at any point $x_0 < a = d - 2u^2$ means that $x_1 > d$, and so that process diverges also:



This leads us to the following statement:

Lemma 3.0.1. *For any secular function of the form* (4)*, and a starting point* $x_0 \notin \langle d - u^2, d\rangle$*, Newton's method diverges.*

We refrain from proving this result rigorously as it is trivially true. We will now prove a stronger statement in the case of a general secular function.

Proposition 3.1. *Let* $u_i, d_1 < d_2 < ... < d_n$ *be arbitrary for each i=1, 2, ..., n. There exists a coefficient* $\rho > 0$ *with the following property:*
Let $f : \mathbb{R} \to \mathbb{R}$ *be a secular function of the form:*

$$f(x) = 1 - \rho \sum_{i=1}^{n} \frac{u_i^2}{d_i - x},$$

Then there exist "critical intervals" $\langle a_i, b_i\rangle$ *such that:*

- $d_i < a_i < b_i < d_{i+1}$*, for each* $i = 1, 2, ..., n-1$*, and*
- *Newton's method diverges for any starting point* $x_0 \in \langle a_i, b_i\rangle$*.*

Proof. First, we define $g : \mathbb{R} \to \mathbb{R}$ as:

$$g(x) = x - \frac{f(x)}{f'(x)}$$

in other words, $x_{k+1} = g(x_k)$ is the next iteration of Newton's method after x_k. Next, we observe the second derivative of f:

$$f''(x) = -\rho \sum_{i=1}^{n} \frac{u_i^2}{(d_i - x)^3}$$

From its asymptotic behavior around the discontinuities d_i, it is clear that f'' has a root x_i' such that $d_i < x_i' < d_{i+1}$, for each $i = 1, 2, ..., n-1$. Note that the values x_i are independent of ρ. Now we define the constants A_i and B_i by the following:

$$A_i = \sum_{j=1}^{n} \frac{u_j^2}{d_j - x_i'}$$

$$B_i = \sum_{j=1}^{n} \frac{u_j^2}{(d_j - x_i')^2}$$

It follows that for each $i = 1, 2, ..., n$

$$f(x_i') = 1 - \rho A_i, \text{ and}$$
$$f'(x_i') = -\rho B_i$$

and finally,

$$g(x_i') = x_i' - \frac{1 - \rho A_i}{-\rho B_i} = x_i' + \frac{1}{\rho}\frac{1}{B_i} - \frac{A_i}{B_i} \tag{5}$$

The expression above tends towards infinity as ρ tends towards 0, and so the first iteration of Newton's method starting at $x_0 = x_i'$ for some i may be arbitrarily large if ρ is sufficiently small. Finally, because f and f' are continuous, we can find numbers $\alpha_1, \alpha_2, ..., \alpha_n > 0$ and $\beta_1, \beta_2, ..., \beta_n > 0$ such that the previous statement is true for any starting point $x_0 \in \langle x_i' - \alpha_i, x_i' + \beta_i \rangle$. □

Remark. *Newton also diverges for a larger class of starting points, such that an iteration starting in one interval $\langle d_i, d_{i+1} \rangle$ winds up in a different "critical interval" $\langle a_j, b_j \rangle$.*

It is difficult to calculate these critical intervals for an arbitrary secular function, and it is for that reason we have chosen to forgo searching for secular roots via Newton's method, and implemented a binary search instead.

4 Testing and conclusion

The implementation works fine with randomly generated matrices of a small order (20x20 or smaller), however problems arise when applied to a larger matrix. Unlike most numeric processes, DAC iterations are not controlled via residual norms. The only step where accuracy is controlled is during the scalar root search, which is relatively "deep" within the algorithm. It is difficult to control final residuals this way, so the accuracy demand is left to a vague guess, which is not very mathematical at all. Furthermore, due to time constraints, we have failed to transfer implementation to a more powerful platform such as CLAPACK. We suspect that the program crashes when faced with a larger matrix due to lack of memory access and/or lack of compilation when run in MATLAB, unfortunately we can't verify that without comparing it to a more stable implementation.

In conclusion, "Divide and Conquer" is an elegant alternative to a common problem in numerical analysis, however we failed to test it rigorously enough to ascertain its efficacy in comparison to other solutions.

References

[1] Nela Bosner. "PHD Lecture on Numerical Analysis". In: 10 (2012).

[2] J.Demmel. *Applied numerical linear algebra*. 1997.