SCIENCES SORBONNE UNIVERSITÉ

Sorbonne Université

Project SFPN
Parallel computing
Report

Parallelization of the resolution of the secular equation in OpenMP

Students :
Boubacar Diallo
Jeffrey-Chris Kebey

Teacher :
Lokmane Abbas Turki

June 10, 2020

Contents

1 Introduction

The eigendecomposition of a matrix is a fundamental problem in sciences, especially in physics and mathematics. In case of symmetric matrices, several decompositions can be used to obtain eigenvalues, eigenvectors. In this project, a Householder tridiagonalization and a eigendecompistion method on the tridiagonal matrix can be used to resolve the problem. For finding the decomposition, we execute a Divide and Conquer method and this method uses secular equations. To resolve them, we need to find roots. In order to do this we use two algorithm : the Gragg algorithm which is monotonous and has a cubic convergence; the hybrid algorithm which is numerically less complex. In this report, we present all these aspects and in particular we detail the two algorithms and compare their performance after a parallelization in OpenMP.

2 Context

In this part, we present the divide and and conquer method, to the divide phase to the secular equation.

A symmetric tridiagonal eigensolver computes the spectral decomposition of a tridiagonal matrix M such that:

$$M = U\Lambda U^T$$

where U are the eigenvectors, and Λ the eigenvalues. The Divide and Conquer algorithm contents three phases :

1. Partitioning the tridiagonal matrix M into p subproblems.
2. Solve the p smaller problems.
3. Merge the subproblems which are defined by a rank-one modification of a tridiagonal matrix to obtain the solution of the problem.

Now, we define $p = 2$, the problem is partitioned in two sub problems and gives :

$$M = \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix} + \rho u u^T$$

where where T_1 and T_2 are the two tridiagonal submatrices, with the first and the last element modified by subtracting ρ, respectively, u is a vector where $u_i = 1$ only when $i = \frac{n}{2}$ or $i = \frac{n}{2} + 1$. If the solution of the eigenproblem of T_1 and T_2 are given by :

$$M_1 = U_1 D_1 U^T$$

and

$$M_2 = U_2 D_2 U^T$$

The third phase consists to combine the two previous equations to solve the system :

$$M = U(D + \rho zz^T)U^T$$

where $z^T = (u_1^T, u_2^T)$. Now, we just have to find the eigenvalues of $D + \rho zz^T$. Let λ a eigenvalue of $D + \rho zz^T$, we can write :

$$det(D + \rho zz^T - \lambda I) = det((D - \lambda I)(I + \rho(D - \lambda I)^{-1} zz^T)$$

This lemma says : if x and y are vectors, $det(I + xy^T) = 1 + y^T x$.

According to this lemma :

$$\begin{aligned} det(I + \rho(D - \lambda I)zz^T) &= 1 + \rho z^T (D - \lambda I)^{-1} z && (2.1) \\ &= 1 + \rho \sum_{i=1}^{n} \frac{\zeta_i^2}{\delta_i - \lambda} && (2.2) \end{aligned}$$

The equation (2.2) is known as the secular equation and to resolve it, we must find λ. In this project, we use two algorithms to find zero : the Gragg method and the hybrid method.

3 Algorithms parameters

In this part, we set algorithm parameters for both algorithms. Before this, according to Gragg and Ren-Can Li, we set the different parameters of the secular equation (2.2).

For all following paragraphs, we define a secular equation :

$$f(\lambda) = 1 + \rho \sum_{i=1}^{n} \frac{\zeta_i^2}{\delta_i - \lambda} \tag{3.1}$$

where every $\zeta_i \neq 0$, $\delta_i < \delta_{i+1}$ and $\rho > 0$. We make $D = (\delta_i, ..., \delta_n)$ and $z = (\zeta_1, \zeta_2, ..., \zeta_n)^T$ with $||z|| < 2$. We choose these values to have a quick and reliable (with a minimum of deflation) convergence. The value of ρ should not be too big (generally $\rho < 100$).
Even if they are different, Gragg and hybrid methods have common phases : the initialization phase and the stopping phase.

3.1 Initialization phase

For finding a value λ_k in an interval $]\delta_k, \delta_{k+1}]$, we need to determinate an initial approximation of λ_k. To have an optimal convergence and avoid several problems like a zero division or a collision between λ_k and δ_k or δ_{k+1}, we determine two things : the value λ_k

more or less close to δ_k or δ_{k+1} and whether $k = n$.
A simple way to make the right decision is to determinate the sign of $f(\frac{\delta_k+\delta k+1}{2})$. If the value is positive, λ_k is closer to δ_k than δ_{k+1} otherwise, λ_k is closer to λ_{k+1}. The value of k is important, if $k = n$, we try to find λ_k in the interval $]\delta_n, \delta_{n+1}]$. For both algorithms, we need to set and compute $\delta_{n+1} = \delta_n + \frac{z^T z}{\rho}$.
We redefine the secular function as $f(x) = g(x) + h(x)$ where,

$$g(x) = \sum_{j=1, j\neq k,k+1}^{n} \rho + \frac{\zeta_j^2}{\delta_j - x} \quad \text{and} \quad h(x) = \frac{\zeta_k^2}{\delta_k - x} + \frac{\zeta_{k+1}^2}{\delta_{k+1} - x} \tag{3.2}$$

In case of $1 < k < n$:
The initial approximation y of λ_k is one of two roots of the following equation :

$$g(\frac{\delta_k + \delta_{k+1}}{2}) + h(y) = 0 \tag{3.3}$$

If $f(\frac{\delta_k+\delta k+1}{2}) \geq 0$, the equation (3.3) can be solved with $\tau = \delta_k + y$ otherwise, the equation is solved with $\tau = \delta_{k+1} + y$. We define $\Delta = \delta_{k+1} - \delta_k$ and $c = g(\frac{\delta_k+\delta k+1}{2})$, and we obtain a formula for τ :

$$\tau = y - \delta_K = \frac{a - \sqrt{a^2 - 4bc}}{2c} \quad \text{if} \quad a \leq 0 \tag{3.4}$$

$$= \frac{2b}{a + \sqrt{a^2 - 4bc}} \quad \text{if} \quad a > 0 \tag{3.5}$$

where $f(\frac{\delta_k+\delta k+1}{2}) < 0$:

$$K = k + 1, a = -c\Delta + (\zeta_k^2 + \zeta_{k+1}^2), b = -\zeta_{k+1}^2 \Delta; \tag{3.6}$$

and where $f(\frac{\delta_k+\delta k+1}{2}) \leq 0$:

$$K = k, a = c\Delta + (\zeta_k^2 + \zeta_{k+1}^2), b = \zeta_k^2 \Delta. \tag{3.7}$$

In case of $k = n$:

The initial approximation y of λ_k is one of two roots of the following equation :

$$g(\frac{\delta_n + \delta_{n+1}}{2}) + h(y) = 0 \tag{3.8}$$

The equation (3.8) can be solved with $\tau = y - \delta_n$ and we obtain the following formulas :

- if $\frac{\delta_n+\delta_{n+1}}{2} \leq \lambda_n$, i.e., $f(\frac{\delta_n+\delta_{n+1}}{2}) \leq 0$.
    1. If $g(\frac{\delta_n+\delta_{n+1}}{2}) \leq -h(\delta n + 1)$, then $\tau = y - \delta_n = \frac{z^T z}{\rho}$.
    2. If $g(\frac{\delta_n+\delta_{n+1}}{2}) > -h(\delta n + 1)$, then :

$$\tau = y - \delta_n = \frac{a + \sqrt{a^2 - 4bc}}{2c} \quad \text{if} \quad a \geq 0 \tag{3.9}$$

$$= \frac{2b}{a - \sqrt{a^2 - 4bc}} \quad \text{if} \quad a < 0 \tag{3.10}$$

where $\Delta = \delta_n - \delta_{n-1}$, $c = g(\frac{\delta_n+\delta n+1}{2})$ and $a = -c\Delta + (\zeta_{n-1}^2 + \zeta_n^2), b = -\zeta_n^2 \Delta$.

- if $\frac{\delta_n+\delta_{n+1}}{2} > \lambda_n$, i.e., $f(\frac{\delta_n+\delta_{n+1}}{2}) > 0$, then $g(\frac{\delta_n+\delta_{n+1}}{2}) >_h (\delta_{n+1})$, we calculate $\tau = y - \delta_n$ exactly with the formulas (3.9) and (3.10).

3.2 Stopping phase

There are several ways to stop Gragg and hybrid algorithms. A simple way is to define a stopping criterion. In Gragg method, when the convergence is monotonous, we can stop the algorithm when :

$$(y_{k-1} - y_{k-2})(y_{k-1} - y_k) \leq 0 \tag{3.11}$$

where y is an approximation of λ_k

When algorithms are not monotonous, we need to determinate another stopping criterion because if we use the previous criteria, algorithms would be too long, even infinite. To guarantee full accuracy of computed distances $\delta_i - \lambda_k$, Ren Cang Li determinate two differnt stopping criteria. The first criterion is defined with :

$$\eta^2 \leq c\epsilon_m min\{|\delta_k - x|, |\delta_{k+1} - x|\}(\eta_0 - \eta) \tag{3.12}$$

where x is the current iterate, ϵ_m the machine's roundoff threshold, c a small constant, η the last correction computed (cf. Gragg algorithm) and η_0 is the correction before last.

According to Gu and Eisenstadt's work, Li determinate a second stopping criterion. He starts to rewrite the secular equation :

$$f(\delta_K + \tau) = 1 + \rho \sum_{j=1}^{n} \frac{\zeta_j^2}{(\delta_i - \delta_K) - \tau} \tag{3.13}$$

where $\tau = x - \delta_K$, $K = k$, when λ_k is close to δ_k and $K = k+1$, when λ_{k+1} is close to δ_{k+1}. He shows that :

$$|f(\delta_K + \tau) - (\text{computed } \ f(\delta_K + \tau))| \leq \epsilon e \tag{3.14}$$

where

$$e = 2\rho + \sum_{j=1}^{k}(k-j+6)|\frac{\zeta_j^2}{\delta_j - x}| + \sum_{j=n}^{k+1}(j-k+5)|\frac{\zeta_j^2}{\delta_j - x}| + |f(\delta_K + \tau)| \tag{3.15}$$

Ren-Cang Li executes recursively the secular equation and calculates the coast of each iteration. He assumes he find τ^* near to τ i.e., $|\tau - \tau^*| \leq \epsilon_m|\tau|$. Li deduces a new stopping criterion :

$$|f(\delta_K + \tau)| \leq e_m + \epsilon_m|\tau||f'(\delta_K - \tau)| \tag{3.16}$$

In our implementation, we choose the criterion (3.16) which is more accurate than the first criterion (3.12).

4 Gragg algorithm

The Gragg algorithm give an approximation of a eigenvalue in an interval indeed, the function can be divide in several intervals $(\delta_1, \delta_2), (\delta_2, \delta_3)...(\delta_n, +\infty)$ and each interval has exactly one zero, each interval admits one λ. In the most part of the intervals, the convergence of the zero finder is monotonous however, in some intervals, the convergence is non-monotonous so, the method has two cases : one where intervals are monotonous and another where intervals are non-monotonous.

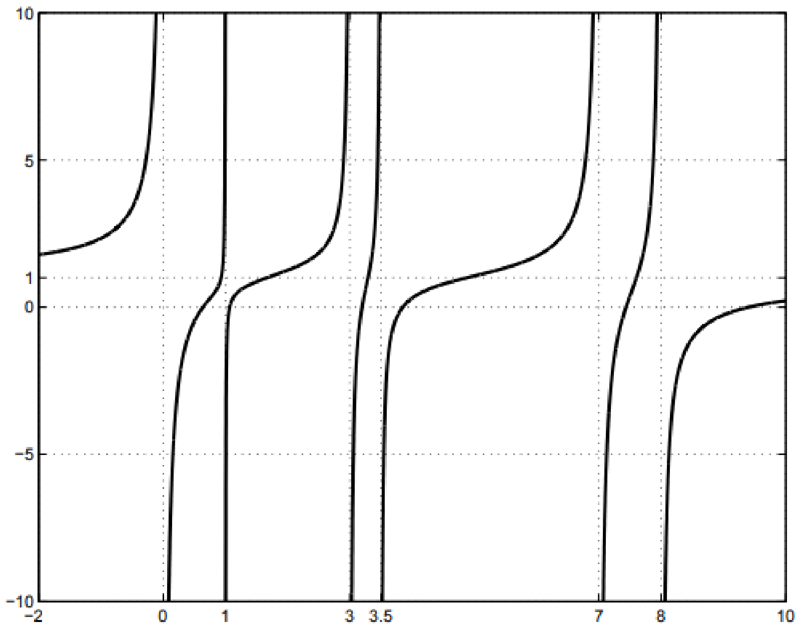

Figure 1: Secular equation with six intervals

To calculate the eigenvalue λ_k in the interval $]\delta_k, \delta_{k+1}]$, we redefine the secular equation :

$$Q(y; c, s, S) = c + \frac{s}{\delta_k - y} + \frac{S}{\delta_{k+1} - y} \tag{4.1}$$

we can rewrite :

$$g(y) = c + \frac{s}{\delta_k - y} + \frac{S}{\delta_{k+1} - y} \tag{4.2}$$

where y is an approximation of δ_k
In this way, we have the following relation with the secular equation (3.1):

$$g(y) = f(y) \tag{4.3}$$
$$g'(y) = f'(y) \tag{4.4}$$
$$g''(y) = f''(y) \tag{4.5}$$

Li deduces a simple way to obtain c,s and S :

$$g'(y) = \frac{s}{(\delta_k - y)^2} + \frac{S}{(\delta_{k+1} - y)^2} \tag{4.6}$$

$$\frac{g''(y)}{2} = \frac{s}{(\delta_k - y)^3} + \frac{S}{(\delta_{k+1} - y)^3} \tag{4.7}$$

and finally, we obtain :

$$s = \frac{\Delta_k^3 \Delta_{k+1}}{\Delta_k - \Delta_{k+1}} (\frac{g'(y)}{\Delta_{k+1}} - \frac{g''(y)}{2}) \tag{4.8}$$

$$= \zeta_k^2 + \frac{(\delta_k - y)^3}{\delta_k - \delta_{k+1}} \sum_{i \neq k, k+1}^{n} \frac{\delta_i - \delta_{k+1}}{(\delta_i - y)^3} \zeta_i^2 > \zeta_k^2 \tag{4.9}$$

$$S = \frac{\Delta_k \Delta_{k+1}^3}{\Delta_{k+1} - \Delta_k} (\frac{g'(y)}{\Delta_k} - \frac{g''(y)}{2}) \tag{4.10}$$

$$= \zeta_{k+1}^2 + \frac{(\delta_{k+1} - y)^3}{\delta_{k+1} - \delta_k} \sum_{i \neq k, k+1}^{n} \frac{\delta_i - \delta_k}{(\delta_i - y)^3} \zeta_i^2 > \zeta_{k+1}^2 \tag{4.11}$$

$$c = g(y) - (\Delta_k + \Delta_{k+1}) g'(y) + \Delta_k \Delta_{k+1} \frac{g''(y)}{2} \tag{4.12}$$

where $\Delta_k = \delta_k - y$ and $\Delta_{k+1} = \delta_{k+1} - y$.

The principle of the Gragg method is to use a "Newton method" liked. At each iteration, we calculate an correction η, we add the correction to the previous approximation y, this number become the new approximation for the next iteration. The algorithm stops when the approximation has attained a stopping criteria. So, we define:

$$\eta = \frac{a - \sqrt{a^2 - 4bc}}{2c} \quad \text{if} \quad a \leq 0, \tag{4.13}$$

$$= \frac{2b}{a + \sqrt{a^2 - 4bc}} \quad \text{if} \quad a > 0, \tag{4.14}$$

where

$$a = (\Delta_k + \Delta k + 1) f(y) - \Delta k \Delta k + 1 f'(y), \tag{4.15}$$

$$b = \Delta_k \Delta k + 1 f(y), \tag{4.16}$$

where $\Delta_k = \delta_k - y$ and $\Delta_{k+1} = \delta_{k+1} - y$.

5 Hybrid algorithm

The hybrid algorithm is an alternation between two methods : the middle way and the fixed weight method. Through a combination of both methods, we obtain more powerful equation solvers. The strength of this algorithm is at some point to use three poles (δ_{k-1}, δ_k and δ_{k+1}) instead of two (δ_k and δ_{k+1}) to speed up the λ determination. The middle way and the fixed weight method use a rewrite of the secular equation :

$$Q(x; c, s, S) = c + \frac{s}{\delta_k - x} + \frac{S}{\delta_{k+1} - x} \tag{5.1}$$

5.1 The middle way

Ren-Cang Li uses a redefinition of the secular equation $f(x) = g(x) + h(x)$ where :

$$g(x) = \sum_{j=1}^{k} \frac{\zeta_j^2}{\delta_j - x} \quad \text{and} \quad \sum_{j=k+1}^{n} \frac{\zeta_j^2}{\delta_j - x} \tag{5.2}$$

The middle way consists of an interpolation of $g(x)$ and $h(x)$ by rationals of types $G(x; \delta, r, s)$. When $0 < k < n$, Li lets :

$$r + \frac{s}{\delta_k - x} \quad \text{approximate to} \quad)g(x) \tag{5.3}$$

and

$$R + \frac{s}{\delta_{k+1} - x} \quad \text{approximate to} \quad)g(x) \tag{5.4}$$

where $\left\{ \begin{array}{lcl} s & = & \Delta_k^2 g'(y) > 0, \\ r & = & g(y) - \Delta_k g'(y) \leq 0, \end{array} \right.$ and $\left\{ \begin{array}{lcl} S & = & \Delta_k^2 h'(y) > 0, \\ R & = & h(y) - \Delta_k h'(y) \leq 0, \end{array} \right.$

where $\Delta_k = \delta_k - y =< 0 < \Delta_{k+1} = \delta_{k+1}$. Li compute an better approximation $y + \eta$ to $_k$ by solving the equation :

$$\rho + r + \frac{s}{\delta_k - x} + R + \frac{S}{\delta_{k+1} - x} = 0. \tag{5.5}$$

The equation has two roots x and we choose the root between δ_k and δ_{k+1}. If $\eta = x - y$, then the correction η is :

$$\eta = \frac{a - \sqrt{a^2 - 4bc}}{2c} \quad \text{if} \quad a \leq 0, \tag{5.6}$$

$$= \frac{2b}{a + \sqrt{a^2 - 4bc}} \quad \text{if} \quad a > 0, \tag{5.7}$$

where :

$$a = (\Delta_k + \Delta k + 1) f(y) - \Delta_k \delta_{k+1} f'(y), \tag{5.8}$$

$$b = \Delta_k \Delta k + 1 f(y), \tag{5.9}$$

$$c = f(y) - \Delta_k g'(y) - \Delta_{k+1} h'(y). \tag{5.10}$$

5.2 The fixed weight method

In some cases, with the middle way , the weight of δ_k and δ_{k+1} can be overestimated. This overestimation makes iteration slower. Ren-Cang Li proposes the fixed weight method. In this method, Li fixes one of the weight δ_k or δ_{k+1} to satisfy the equation (4.2) and (4.6).

- If λ_k is closed to δ_k, we set $s = \zeta_k^2$, then :

$$S = \Delta_{k+1}^2(f'(y) - \frac{\zeta_k^2}{\Delta_k^2}) \tag{5.11}$$

$$= \zeta_{k+1}^2 + \sum_{j \neq k,k+1}^{n} \frac{\Delta_{k+1}^2}{\Delta_j^2}\zeta_j^2 > \zeta_{k+1}^2 \tag{5.12}$$

$$c = f(y) - \frac{\zeta_k^2}{\Delta_k} - \Delta k + 1(f'(y) - \frac{\zeta_k^2}{\Delta_k^2}) \tag{5.13}$$

$$= f(y) - \Delta_{k+1}f'(y) - \frac{\zeta_k^2}{\Delta_k^2}(\delta_k - \delta_{k+1} \tag{5.14}$$

- If λ_k is closed to δ_{k+1}, we set $S = \zeta_{k+1}^2$, then :

$$s = \Delta_k^2(f'(y) - \frac{\zeta_{k+1}^2}{\Delta_{k+1}^2}) \tag{5.15}$$

$$= \zeta_k^2 + \sum_{j \neq k,k}^{n} \frac{\Delta_k^2}{\Delta_j^2}\zeta_j^2 > \zeta_k^2 \tag{5.16}$$

$$c = f(y) - \Delta_k f'(y) - \frac{\zeta_{k+1}^2}{\Delta_{k+1}^2}(\delta_{k+1} - \delta_k) \tag{5.17}$$

5.3 Hybrid scheme

For $m = 2, ..., n-1$, we define a secular function f(x) with the mth term in the summation removed :

$$f_m(x) = \rho + \sum_{j=1, j \neq m}^{n} \frac{\zeta_j^2}{\delta_j - x} \tag{5.18}$$

The function has a zero between δ_{m-1} and δ_{m+1} and Ren-Cang Li finds two difficult possibilities :

- $\delta_k < \lambda_k < \frac{\delta_k + \delta_{k+1}}{2}$ and $f_k(x)$ has a zero between δ_k and λ_k
- $\frac{\delta_k + \delta_{k+1}}{2} < \lambda_k < \delta_{k+1}$ and $f_{k+1}(x)$ has a zero between λ_k and δ_{k+1}

The treatment of the two cases is almost the same. In what follows, we treat the first case only. We can interpolate $f(x)$ with the following rational :

$$\tilde{Q}(x; c, s, S) = c + \frac{s}{\delta_{k-1} - x} + \frac{\zeta_k^2}{\delta_k - x} + \frac{S}{\delta_{k+1} - x} \tag{5.19}$$

The parameters c, s and S are determined with the middle way method or the fixed weight method. But, in some cases, we cannot compute $\tilde{Q}(x; c, s, S)$, because of roundoff, f(y) and $\tilde{Q}(x; c, s, S)$ are clearly different even through we might be the same in theory. If computed $f(x)$ is available at the time we interpolate the secular equation, we do not compute $\tilde{Q}(x; c, s, S)$ at while setting it to be $f(y)$. If the next time we must compute $\tilde{Q}(x; c, s, S)$, we add a correction and update $f(y)$ because :

$$\tilde{Q}(x; c, s, S) = \tilde{Q}(y; c, s, S) + (\tilde{Q}(x; c, s, S) - \tilde{Q}(y; c, s, S)) \tag{5.20}$$

$$= f(y) + (x+y)(\frac{s}{(\delta_{k-1}-y)-(x-y)} + \frac{\zeta_k^2}{\delta_{k-1}-y)-(x-y)} + \frac{S}{(\delta_{k+1}-y)-(x-y)}). \tag{5.21}$$

The evaluation of $\tilde{Q}(x; c, s, S)$ is done in the same way by adding correction to its value at the previous x.

Ren-Cang Li gives us an example of hybrid method operation. We suppose we are computing λ_k in case $\delta_k < \lambda_k < \frac{\delta_k+\delta_{k+1}}{2}$. We have an initial guess $\delta_k+\tau$ for which $\delta_k < \delta_k+\tau < \lambda_k$ is guarantee. We obtain a secular function :

$$f(\delta_k+\tau) = f_k(\delta_k+\tau) + \frac{\zeta_k^2}{-\tau} < 0 \tag{5.22}$$

We make a decision between using two poles or three poles : if $f(\delta_k+\tau) > 0$, we use two poles δ_k and δ_{k+1} otherwise, we use three poles δ_{k-1}, δ_k and δ_{k+1}. After the choice, for the first iteration, we use the fixed eight method to interpolate $f(x)$. If the decision was three poles, we do anther iteration with the fixed weight method to get a new approximation τ_1. If $f(\delta_k+\tau_1) < 0$ and $|f(\delta_k+\tau_1)| > 0.1*|f(\delta_k+\tau)|$, we switch to the middle way method and the next iteration starts with $\delta_k+\tau_1$. After that, at each iteration, we compare the value $fnew$ of the secular function for the new approximation to $fpre$, the value of the secular function for the previous approximation. We switch the methods when $fnewfpre > 0$ and $|f_{new}| > 0.1 * |f_{pre}|$. The algorithm stops when a stopping criterion is attained.

6 Tests and performance

The timing runs were performed on the ppti-gpu-1 at Sorbonne University(campus Jussieu).This computer is equipped with 44 cores Intel(R) Xeon(R) Gold 6152 CPU(2.10GHz) and 384GB RAM.
To test the performance of our two implementations and compare their results, we need to define all data. We determinate $D = (\delta_1, \delta_2, ..., \delta_n)$ where each δ is chosen uniformly randomly and ordered in this way : $\delta_1 < \delta_2 < ... < \delta_n$, $z^T = (\zeta_1, ..., \zeta_n)$ where each ζ is chosen uniformly randomly and $||z^T|| < 2$, $0 < \rho < 100$ and $0 < n \leq 100000$.
Figure 2 shows the execution time of Gragg and hybrid on different sizes

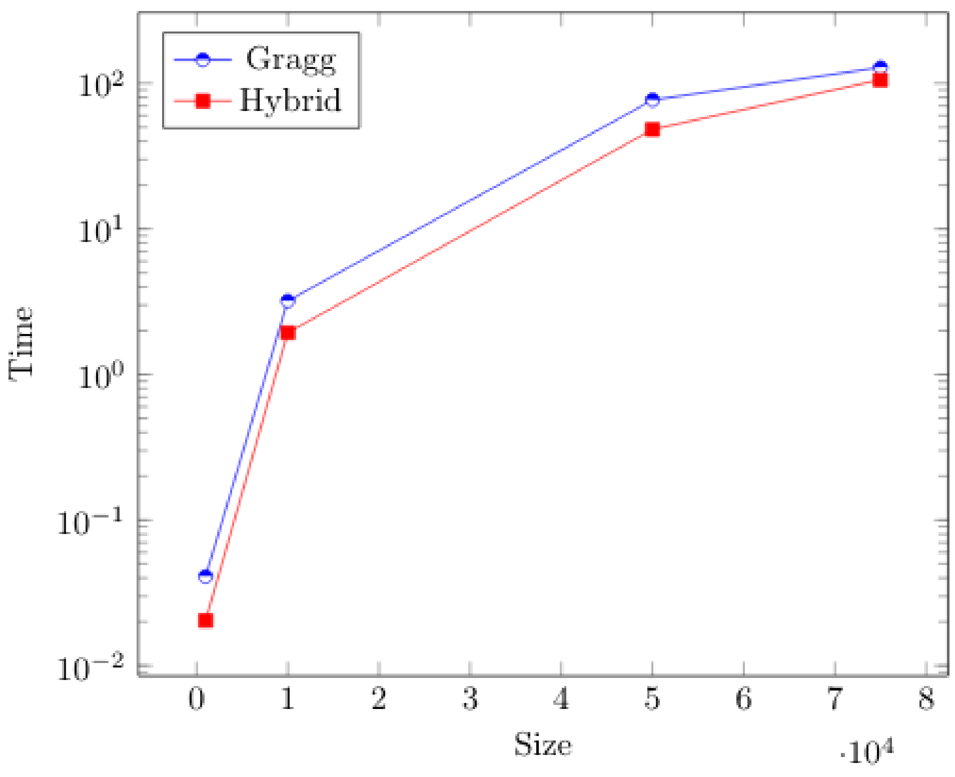

Figure 2: Execution time of Gragg and Hybrid

In this graph, Gragg and Hybrid methods are executed with two threads indeed, we want to obtain the shortest time possible and the speed-up is almost optimal for 2 two threads. Moreover, the number of threads has only a little influence on the difference of time between Gragg and Hybrid. We note Hybrid algorithm is slightly faster than Gragg method indeed, the addition of the third poles increases the speed of solving and particularly, the hybrid method uses only one derivative to find λ_k while Gragg algorithm uses two.

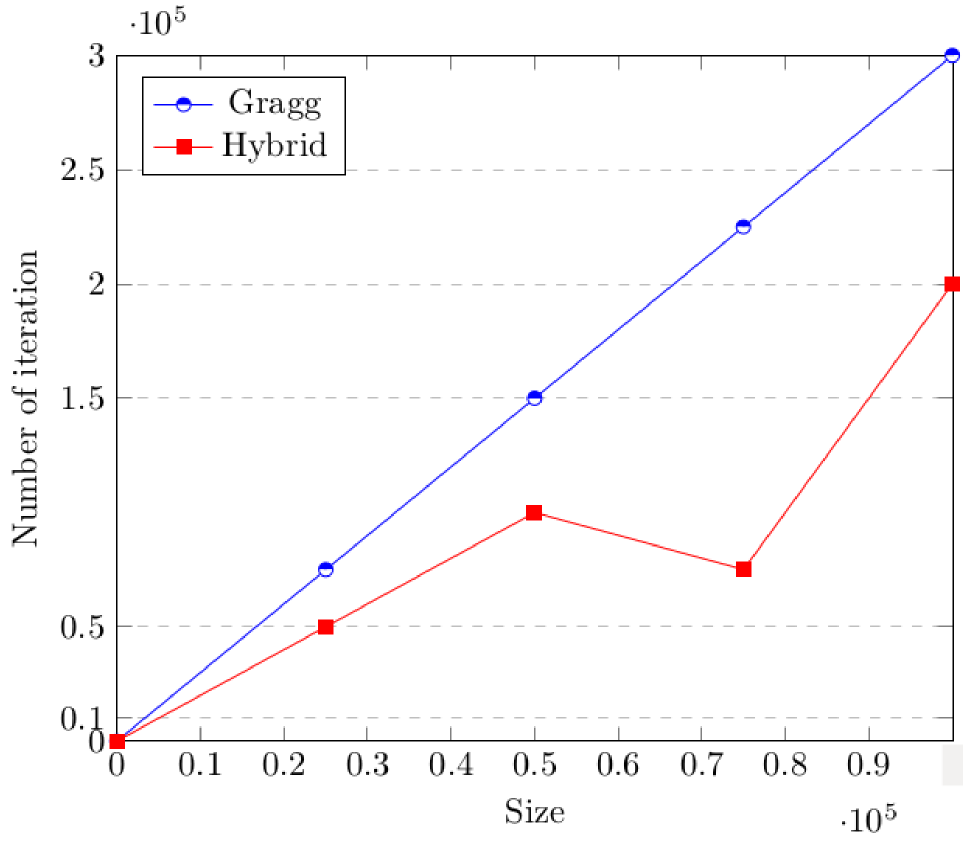

Figure 3: Number of iterations using Gragg and hybrid

This graph shows the total number of iterations (for an algorithm, all iterations to find λ_k are added) in function of the size of a matrix. It confirms the previous deduction, the Hybrid algorithm has less iterations than Gragg method. Hybrid method has more efficient iterations because of the number of derivatives uses by both algorithms (one for hybrid, two for Gragg). The combination between the fixed weight method and the middle way permits hybrid method to reduce the number of iterations and optimise the time of finding λ_k.

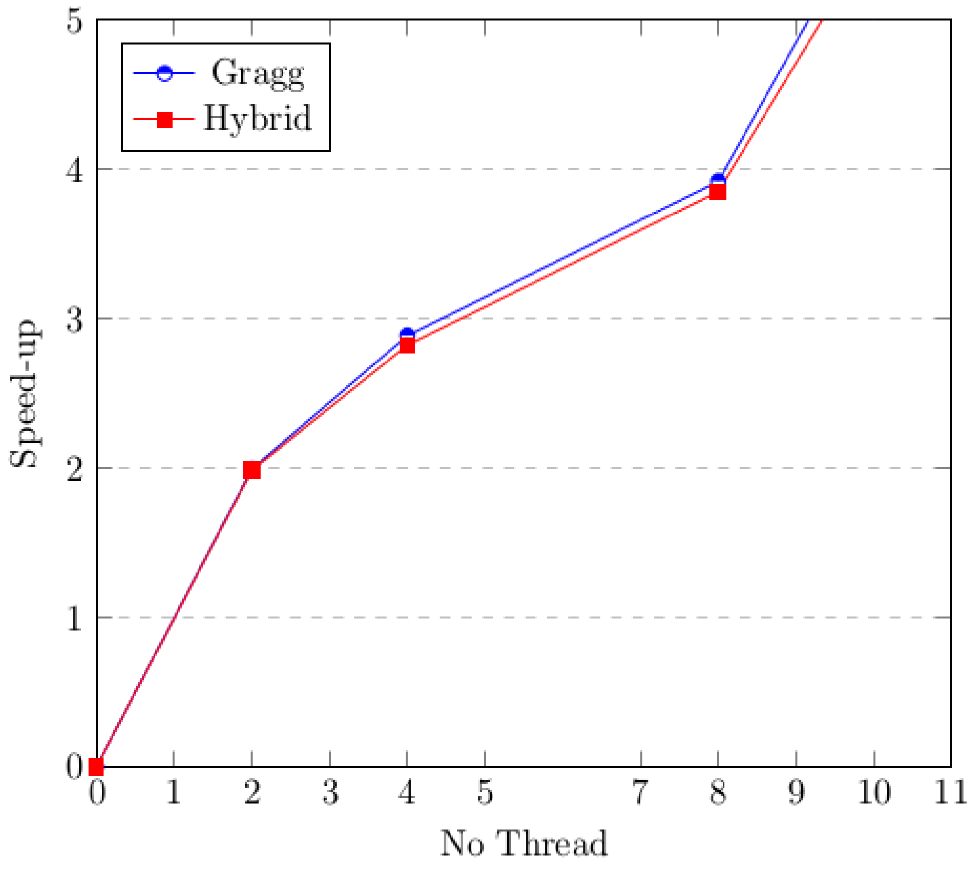

Figure 4: Acceleration of Gragg and Hybrid depending on the number of threads

This figure shows the speed-up of both methods with an OpenMp parallelization. We can see the speed-up is optimal when the number of thread is two, that's why we use two threads to have the best possible result. The acceleration depends on the way we parallelize the program. In this project, we use the command "schedule" from OpenMP.

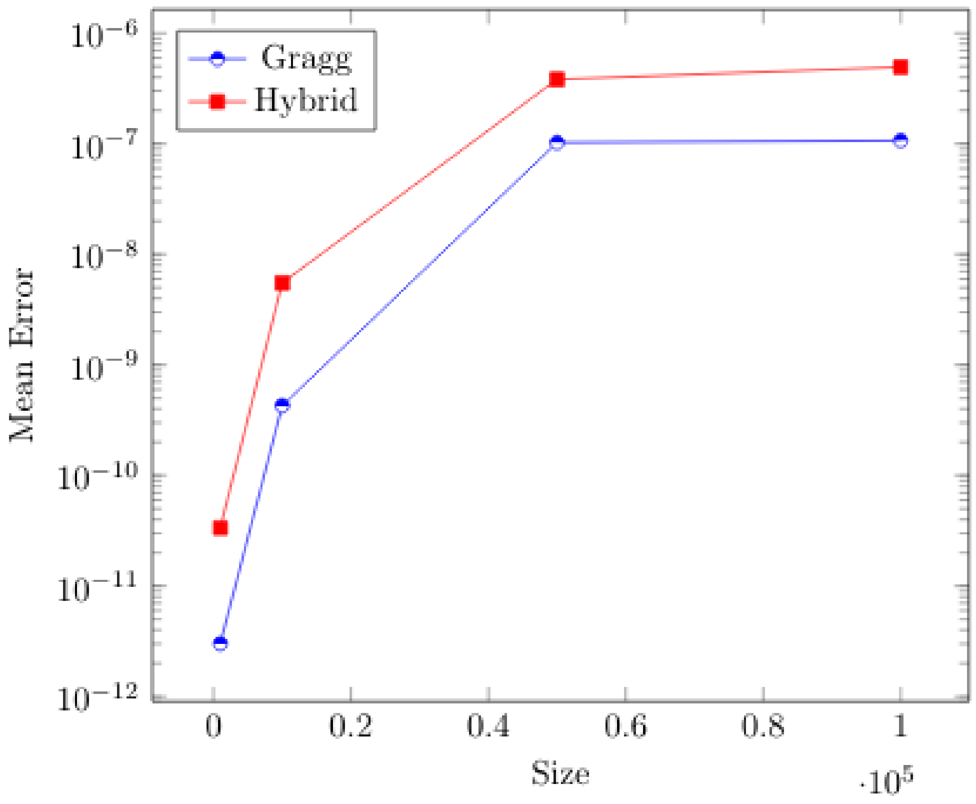

Figure 5: Mean error of Gragg and Hybrid in evaluation of f

This figure shows the precision of the eigenvalue approximation for each algorithm. We note algorithms are more accurate with small matrices than large matrices. Also, we note Gragg algorithm precision is better than Hybrid method. The precision can influence the speed of an algorithm indeed, in general, a less accurate algorithm finishes faster than a more precise method. The precision contributes to the difference of speed between hybrid and Gragg methods.

7 Conclusion

Hybrid and Gragg algorithms are both excellent methods to resolve the secular function problem. However, the hybrid algorithm is slightly better than Gragg algorithe thm, indeed the difference of convergence (cubic for Gragg and quadratic for Hybrid) find in our results again. Our implementation choice (especially the choice of stopping criteria, type of parallelization) may be a source of debate. Therefore, it would be interesting to compare these two methods with other methods of secular equations solving.

References

[1] L. A. Abbas-Turki and S. Graillat, "Resolving small random symmetric linear systems on graphics processing units", *. Supercomput.*, vol. 73, no 4, p. 1360-1386, 2017.

[2] J. Demel, "Applied numerical linear algebra", *SIAM*, 1997.

[3] W. B. Gragg, J. R. Thornton, and D. D. Warner, "Parallel divide and conquer algorithms for the symmetric tridiagonal eigenproblem and bidiagonal singular value problem", *Modeling and Simulation*, 1992.

[4] R.-C. Li, "Solving secular equations stably and efficiently", CALIFORNIA UNIV BERKELEY DEPT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCES, 1993.

[5] C. Vömel, S. Tomov and J. Dongarra, "Divide Conquer on Hybrid GPU-Accelerated Multicore Systems", *SIAM Journal on Scientific Computing*, vol. 34, no 2, p. C70-82, 2012.

[6] W. N. Gansterer, J. Schneid, and C. W. Ueberhuber, "A Divide-and-Conquer Method for Symmetric Banded Eigenproblems-Part I: Theoretical Results", 1998.

[7] A. Melman, "A numerical comparison of methods for solving secular equations", *J. Comput. Appl. Math*, vol. 86, no 1, p. 237-249, 1997.