Project : Parallel implementation and analysis of a new, lowest eigenvalue finding algorithm for possible application in Quantum Chemistry

Paper to be followed throughout discussion

[1] Paper analyzed : J. Chem. Theory Comput., 2015, 11 (2), pp 472–483

What is the thesis of your project? What is the driving science question, issue, software problem?

Scientific problems are often translated into Linear Algebra and thus solving the eigenvalue equation is a common step in multiple areas. Specifically in electronic structure calculations in chemistry and physics, common theories such as DFT, Hartree Fock, Configurational Interaction etc deal with this problem. Since, solution to eigenvalue problem is a $O(N^3)$ step if you require to have all the eigenvalues, it often is the most time consuming step in the calculation.

A central task to electronic structure theory is the solution of Schrodinger equation as an eigenvalue problem.

$$H\Psi=E\Psi \tag{1}$$

where H is the Hamiltonian of the system, E is the set of eigenvalues or Energies and Ψ -> eigenvectors of the wavefunctions of the system. These calculations are used to describe the quantum scale properties of atoms, molecules to materials and surfaces. Here Ψ is usually described by inserting a set of basis functions as :

$$\Psi = \sum c_i \phi_i \tag{2}$$

This result in a generalized matrix eigen problem :

$$H\, c_i = \epsilon_i\, S\, c_i \tag{3}$$

where S is the overlap matrix storing the overlap of each orbital with others and set of c_i are the coefficients of atomic orbitals to be optimised along with ϵ_i. I have kept the H as general H as the steps till (3) are common in many theories. We use the variational principle to iteratively find the solution to (3) to find the minimum energy and wavefunction associated to it. In each iteration, the

solution step involves a $O(n^3)$ order for finding the eigenvalue and eigenvectors.

If we analyse the order of N in the case of our theories, it quickly approaches a far larger number than we can store in our memories. For a CH_2 IF molecule and including d orbitals, $n \sim 10^4$ and makes it harder to do calculations for larger molecules. So there is a need to reduce the order of these calculations as well as parallelism to expand our reach of chemical phenomenon.

A way to get around this problem in some of the theories including DFT and Configurational Interaction, also the newer theories such as Multi-reference Configurational Interaction, is to only deal with the lowest or first few low lying eigenvalues. Algorithms used are Davidson and Davidson Liu algorithms for this purpose. These are efficient but lack the ability to be parallelized properly.[1]

I am presenting a newly published [1] algorithm in this area, analysing it for speedup performances to check and understand possibility for expanding and applying it to the present theories to make such calculations faster and reach bigger, more complex systems to further our understanding of the world.

What did you do? Describe the activities that were undertaken.

Linear Algebra and Chemistry- I read the paper and developed a mathematical understanding of the published solution. I discussed the chemical theories of interest and the possibility to implement and impact of such an algorithm with my research advisor. (Please find the details of the algorithm in the paper[1], the report is strictly based on parallel computing aspects)
Check applied Packages - I researched about the area to understand how and to what extent, the parallel solutions are implemented in popular packages such as LAPACK, BLAS, numpy etc. I came to a conclusion that although basic steps of the calculations are parallelized to a large extent, parallelizing eigenvalue problem/diagonalization of a matrix is still a research topic. Papers by the instructor (Dr. Burns) and discussion with Dr. Da Zheng were very helpful.
Applied Single Processor Code : I developed a working code for single processor first to check its accuracy and error percentage, to consider it's validity.
Parallelized by OpenMp : I first parallelized the code through OpenMp but did not achieve any parallelism. I expect it is due to the complex structure of memory requirements of the two loops in the algorithm.
Parallelized by MPI : Achieved good parallel performance as discussed later.
Analyzed Speedup.

NOTE : To check for the performance of the algorithm, a diagonal dominant form of martix was used (as if often produced in chemical systems) where each element is defined by :

$$A_{ij} = \begin{cases} \frac{-1}{2i+1}, & i=j \\ \frac{-1}{10\,(i+j+1)}, & i \neq j \end{cases}$$

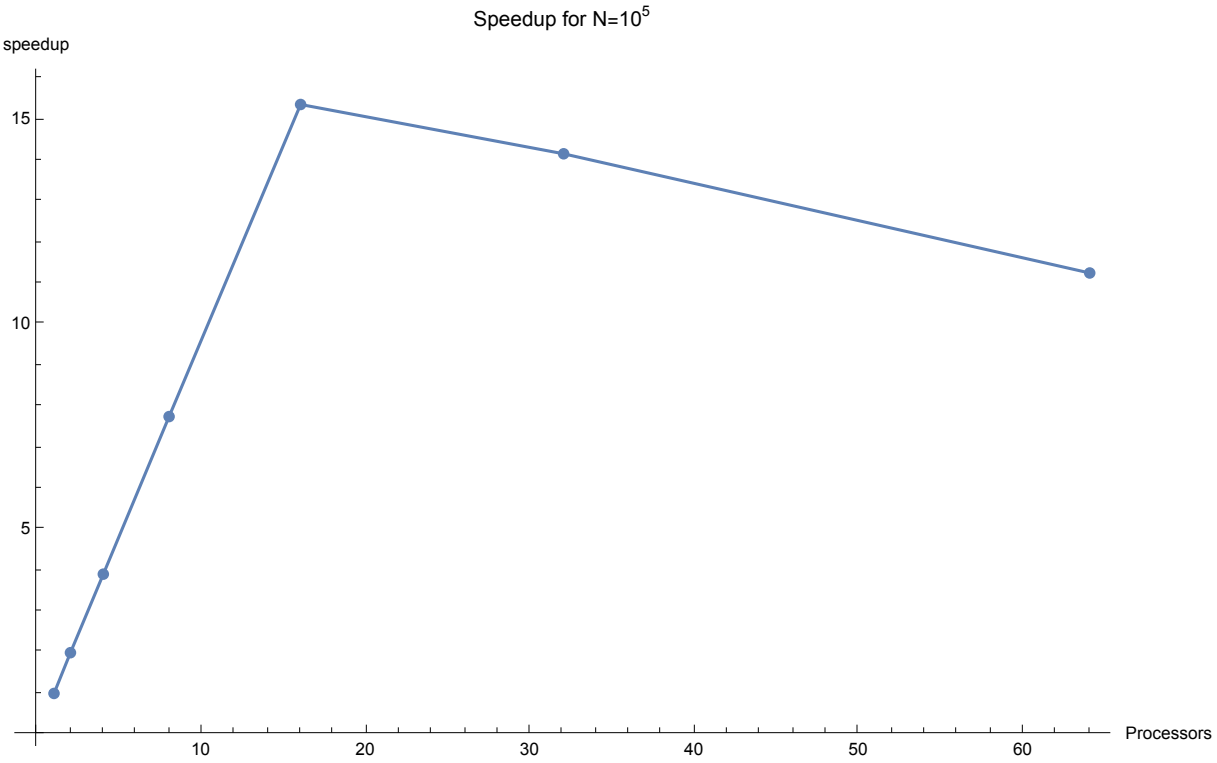
What did you find? Does it support or refute the thesis of the project.

Step 1 : Accuracy check. The eigenvalues generated by the algorithms seem to be in agreement to the required accuracy. For instance, Mathematica produced -1.003729140269 whereas, the algorithm produced -1.0037291 for accuracy till 7th decimal digit for a 2X2 matrix. Another point to note that although this is an iterative algorithm, the number of steps to produce results were ~6 for $N = 10^6$ matrix.

Speedup : In the MPI implementation, amazon aws C4.8 compute node was used for the calculation.

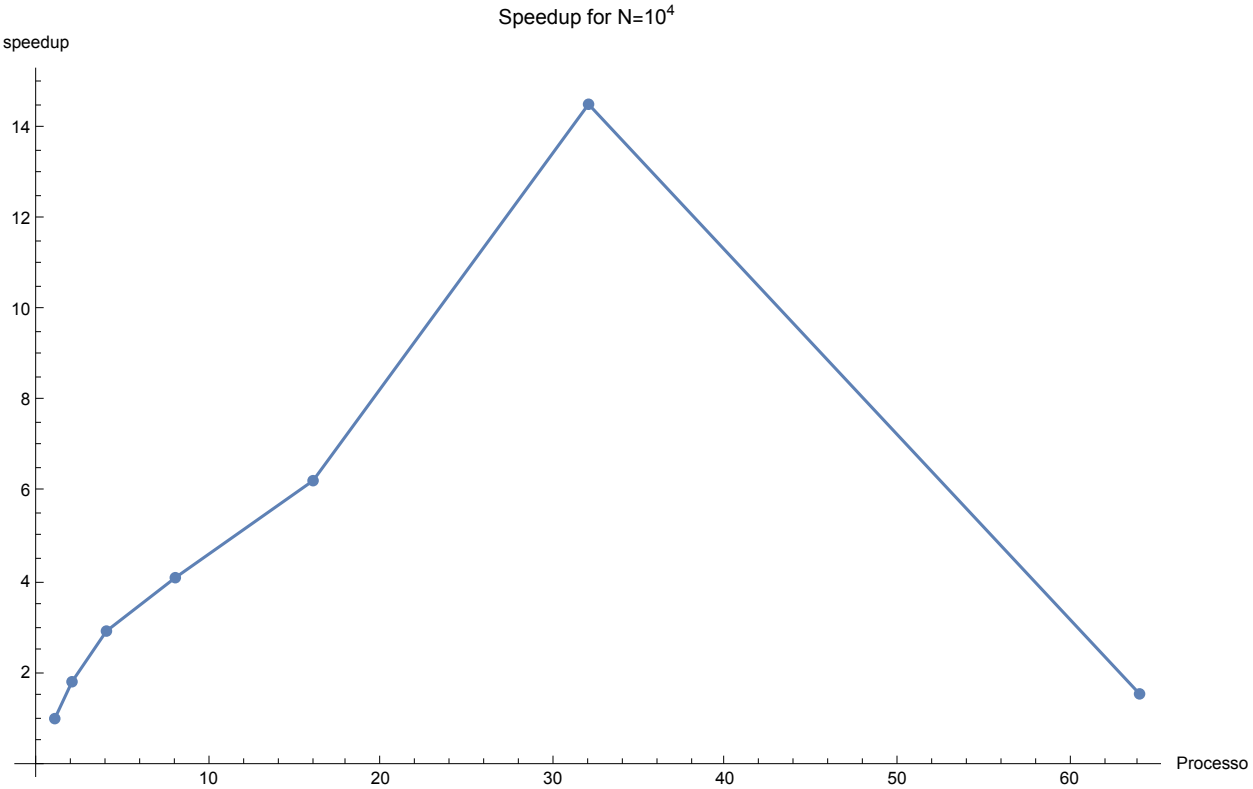
This machine had 36 vCPUs.The speedup produced for $N = 10^5$matrices is shown :

Out[37]=



As can be seen, the speedup is linear (ideal) till 16 vCPUs and decay after that. Interesting observation can also be made for speedup analysis for $N = 10^4$matrices.

Out[38]=



The speedup for lower dimension matrices if fairly bad. This is because of the **fine granularity** or less ratio of computation to communication. The algorithms is : [1]

In[30]:=
```
Import["/Users/ayush/Desktop/algo.png"]
```

Out[30]=

```
alpha=0.d0
alpha0=A(0,0)
iter=0
do while abs(alpha-alpha0).gt.threshold
   alpha0=alpha
   Ap00=A(0,0)
   Ap0i=A(0,i)
   Apii=A(i,i)
   do i=N-1,1,-1 ! Inverse order
      if (iter.ne.0) then
         Delta0i=0.d0
         do j=1,N-1
            if (j.ne.i) Delta0i=Delta0i+A(i,j)*
         end do
         Ap00=alpha0-A(0,i)*c(i)-Delta0i*c(i)
         Ap0i=Ap0i+Delta0i
      end if
      ...
      ! resolution of the 2x2 dressed matrix
      ! to obtain c(i)
      ...
      alpha=alpha+c(i)*A(0,i)
   end do
   iter=iter+1
end do
```

There is an outer loop and inner loop, we parallelize the outerloop and send part of c[i] (eigenvector) to each compute node. If N is small, the task at hand of each compute node is rather small and the communication is large, thereby much time is spent in sending messages in case of $N \sim 10^4$. This is not a problem as our interests are in matrices of much bigger sizes than 10^6. The **skew** is another reason of

bad performance of smaller matrices. Since the parent node is assembling and broadcasting the c[i] matrix, it is essentially doing O(N) work after each iteration of the outer parallelized loop, thus producing a large skew in small matrices.

The speedup produced by $N = 10^5$matrices is excellent till 16 vCPUs and the paper claim better paralelisation for larger matrices (**Course Granularity**). On a computational standpoint the algorithm stands tall and shows potential. There are following tests to be made to analyse the importance of this new algorithm :
1. It needs to be tested as opposed to the more popular implementation such as Davidson-Liu Algorithm.
2. Development on mathematical framework for more number of eigenvalues.

What principle in parallel computing does this project explore?

The project analyzed at tested the **speedup** produced by a MPI implementation of a newly proposed algorithm with possible direct applications in Quantum Chemical theories for chemical properties of atoms, molecules and materials. The project attempted to study scaleup which is important for research problem at hand, but failed due to time and resources limitations.
The project also explored the Granularity of algorithm which suggests that it is an apt algorithm for efficient implementation and performance for large matrices.