Inverse power shift method

Algorithm

Problem : To find an eigen vector whose eigen value is close to our desired number(s)

Solution :

- Assume vector q = c1v1+c2v2+.....cnvn
- Subtract number ‘s’ from matrix A.
- Multiply q with A-1 and store it in q.
- Iterate until ||qupd-q|| < 1e-6.
- Resulting q is our desired eigen vector

How did we parallise those for loops?

In our Code more than 95% of the execution time is taken by the LU decomposition.So if we had mainly focused on parallelizing this for loop.

Step-1: Optimized LU decomposition by removing redundant loops.(Calculating multiplier)

Step-2: As there is dependency between first and second for loop in LU decomposition we parallelized second for loop keeping first variables as shared

Step-3: Used Dynamic Scheduling over Static

```
for(k=1;k<=N-1;k++){
        long long row,col;
        #pragma omp parallel for shared(k) private(row,col) schedule(dynamic)
        for(row=k+1;row<=N;row++){
                double factor = A[row][k]/A[k][k];
                for(col=k+1;col<=N;col++){
                        A[row][col]=A[row][col]-factor*A[k][col];

                }
                A[row][k]=factor;
        }
}
```

RESULTS

BASED ON NUMBER OF CORES :

For N=3000 , number of iterations =1000000,s=-1,default number of threads = 24 .

1 . No of cores = 1

Time taken = 29 sec

2. No of cores = 4

Time taken = 15 sec

RESULTS

BASED ON NUMBER OF CORES :

For N=3000 , number of iterations =1000000,s=-1,default number of threads = 24 .

3 . No of cores = 6

Time taken = 10.5 sec

4. No of cores = 10

Time taken = 7 sec

RESULTS

BASED ON NUMBER OF CORES :

For N=3000 , number of iterations =1000000,s=-1,default number of threads = 24 .

3 . No of cores = 15

Time taken = 5 sec

4. No of cores = 24

Time taken = 3 sec

RESULTS

BASED ON NUMBER OF Threads :

For N=3000 , number of iterations =1000000,s=-1,number of cores = 6 .

1 . No of Threads = 1

Time taken = 30 sec

2. No of Threads = 3

Time taken = 15 sec

RESULTS

BASED ON NUMBER OF Threads :

For N=3000 , number of iterations =1000000,s=-1,number of cores = 6 .

3 . No of Threads = 6

Time taken = 9.5 sec

4. No of Threads = 12

Time taken = 10.5 sec

OBSERVATION

From the above results we can see that,

1.Time taken is least when number of threads used = number of cores or atleast a multiple of it.

2.Time taken decreases as the number of cores increases when number of threads is fixed

3. Time taken decreases and then increases for fixed number of cores and varying the number of threads in the ascending order.