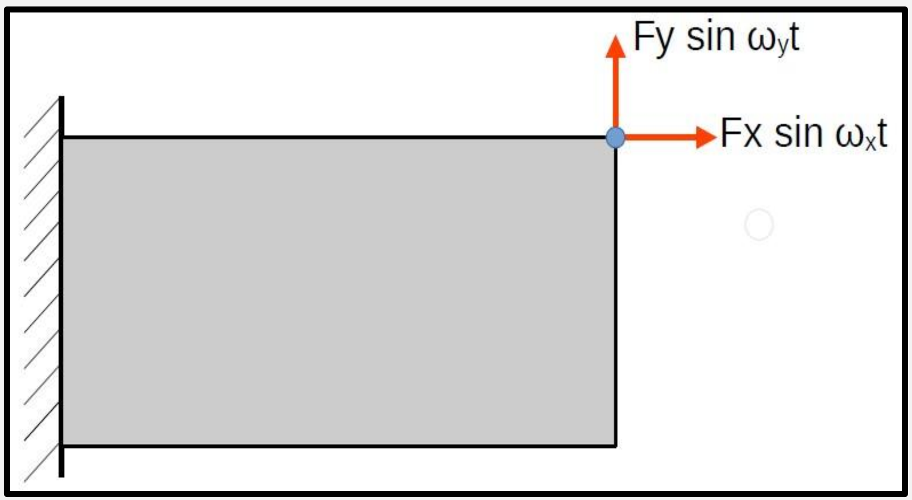
Parallelization of 0th order Generalised Mode Acceleration Method



Find response in x direction of the node at which force is applied

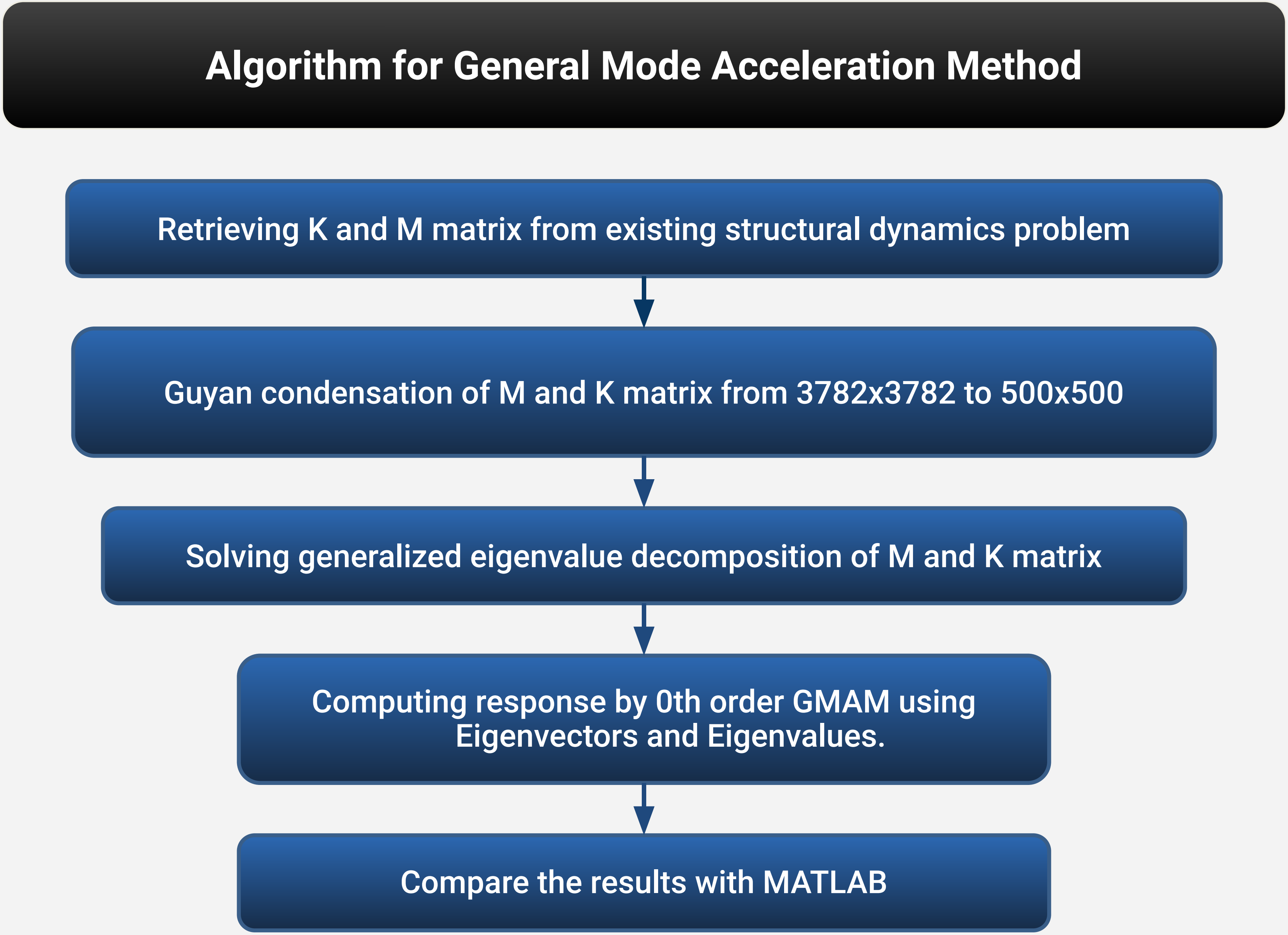
Submitted by:

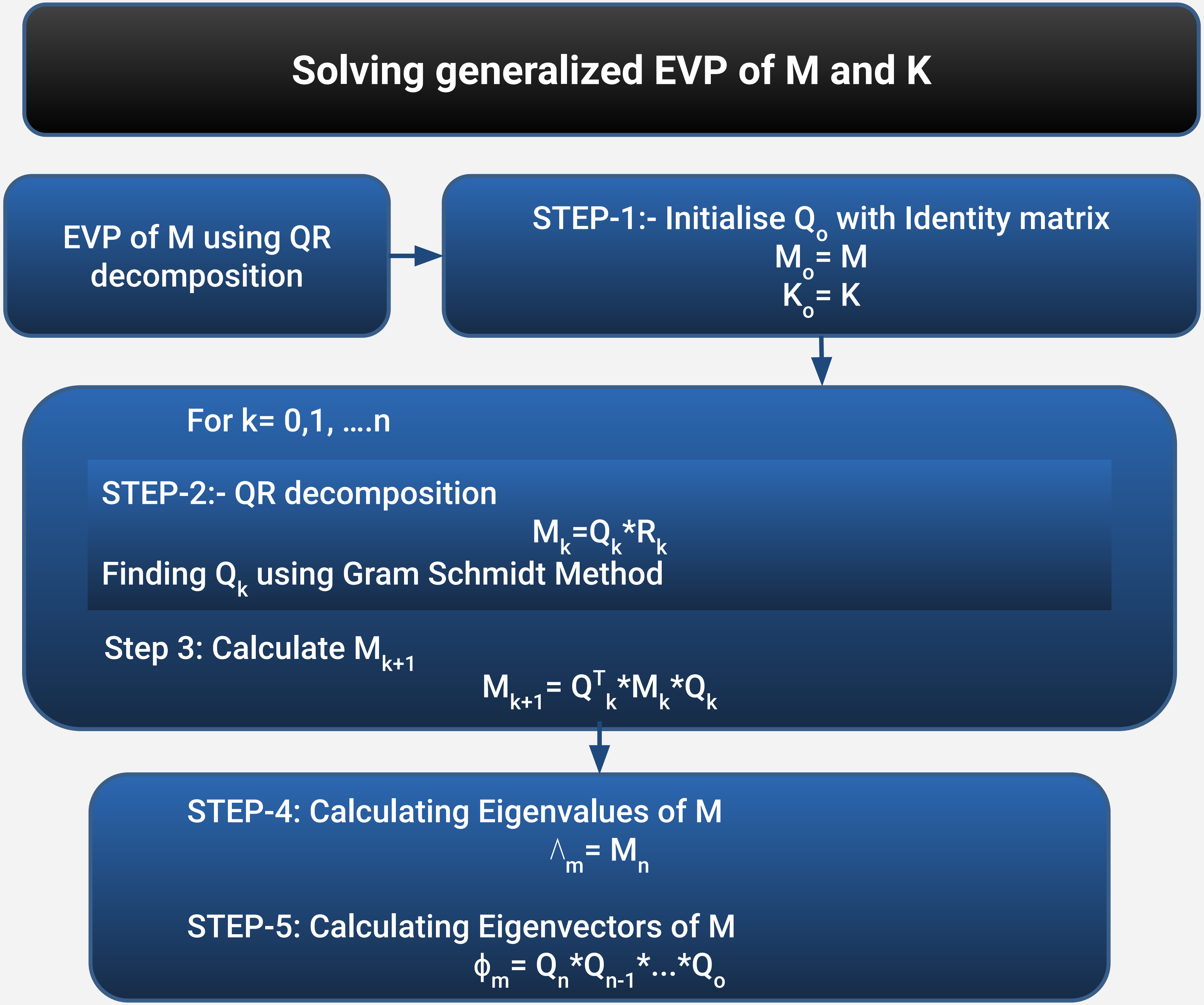
Abhinandan Kumbhar (193109007)

Jalaj Gupta (19310R002)

R. Nitin Iyer (19310R001)

Saurabh Pal (193104001)


Algorithm for General Mode Acceleration Method
Retrieving K and M matrix from existing structural dynamics problem
Guyan condensation of M and K matrix from 3782x3782 to 500x500
Solving generalized eigenvalue decomposition of M and K matrix
Computing response by 0th order GMAM using Eigenvectors and Eigenvalues.
Compare the results with MATLAB


Solving generalized EVP of M and K
EVP of M using QR decomposition
STEP-1:- Initialise Q_o with Identity matrix
$M_o = M$
$K_o = K$
For k= 0,1, ….n
STEP-2:- QR decomposition
$M_k = Q_k * R_k$
Finding Q_k using Gram Schmidt Method
Step 3: Calculate M_{k+1}
$M_{k+1} = Q^T_k * M_k * Q_k$
STEP-4: Calculating Eigenvalues of M
$\Lambda_m = M_n$
STEP-5: Calculating Eigenvectors of M
$\phi_m = Q_n * Q_{n-1} * \ldots * Q_o$

contd....

STEP-7 :- Calculating transformation matrix $\phi(tr)$

$$\phi_{tr} = \phi_m * \wedge_m^{-1/2}$$

STEP-8 :- Transform K with the help of ϕ_{tr}

$$K_{tr} = \phi_{tr}^T * K * \phi_{tr}$$

STEP-9 : Repeat step 2 to 6 for Eigenvalue decomposition of K_{tr} matrix

STEP-10 : Calculating eigenvalues and eigenvectors of generalized eigenvalues of K and M

$$\wedge = K_{trn}$$

$$\phi = \phi_{tr} * \phi_{ktr}$$

MPI Parallelization

QR decomposition parallelization

For k=1…..n

Construction of k^{th} column of Q matrix :-

$Temp_k = A_k - \sum (A_k.Q_n)Q_n$ n = 0…..(k-1)

$Q_k = Temp_k / sqrt(Temp_k.Temp_k)$

Formation of Q parallelized by distribution of $\sum (A_k.Q_n)Q_n$ among the processors.

- Each process has to perform computation of $\sum (A_k.Q_n)Q_n$ for k/cores columns.
- Root process will compute $Temp_K$ and Q_k.

Further procedures

- Lot of matrix multiplication was involved in algorithm.

All the procedures use Matrix multiplication, which can be parallelised.

MPI Parallelization

Matrix multiplication parallelization C=AB :-

- Share B matrix with all the processors.

- Share chunks of rows of A with other processors, and remaining rows will be calculated by A.

  slots=floor(rows of A/ncores), is the chunk of rows being sent to other processors.

- Receive the computed rows of C from other processors and assemble them in C at appropriate positions.

CUDA Parallelization

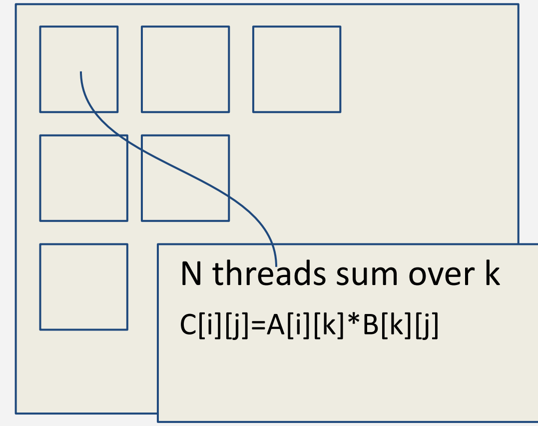
Parallelizable portions of codes are

1. **Matrix multiplication**
2. **QR decomposition**

Matrix multiplication (AB=C) (Each matrix of size N*N)

matmult<<< grid(N,N) , N >>>(A,B,C)

- This kernel launches NxN blocks with N threads in each block
- Each block calculates one entry of C matrix C[i][j]

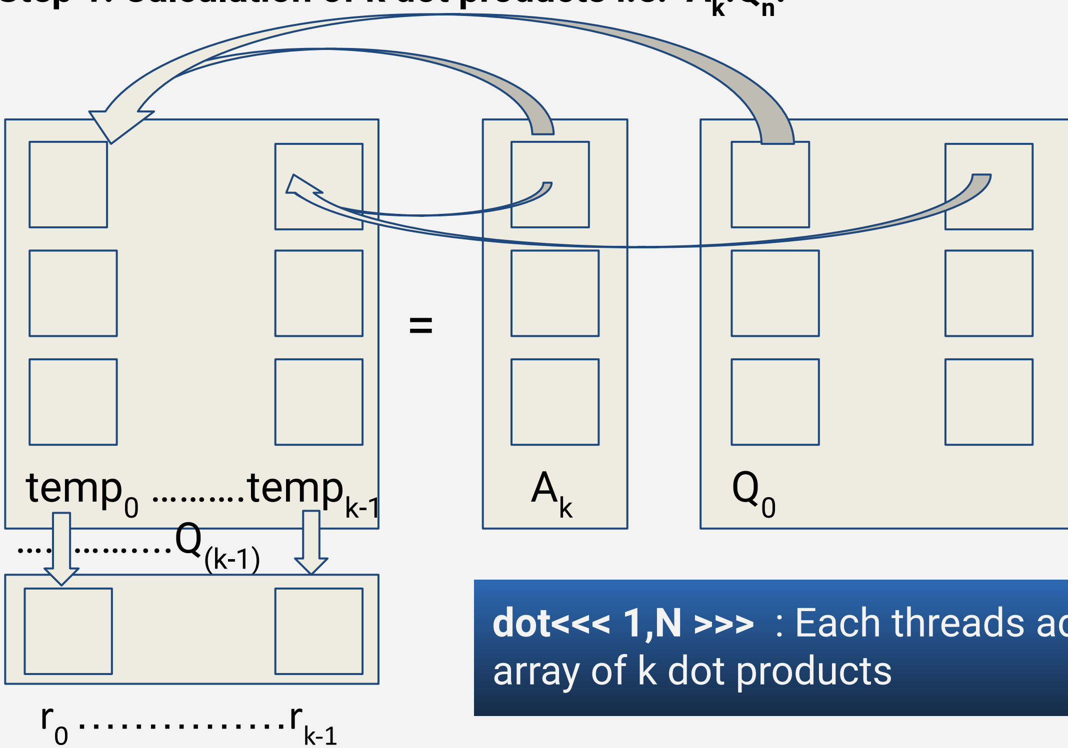


QR decomposition (for calculating k_{th} column of Q_k)

$Temp_k = A_k - \sum (A_k.Q_n)Q_n$ n = 0.....(k-1)   i.e k dot products

$Q_k = Temp_k / sqrt(Temp_k.Temp_k)$

Step 1: Calculation of k dot products i.e. $A_k.Q_n$.



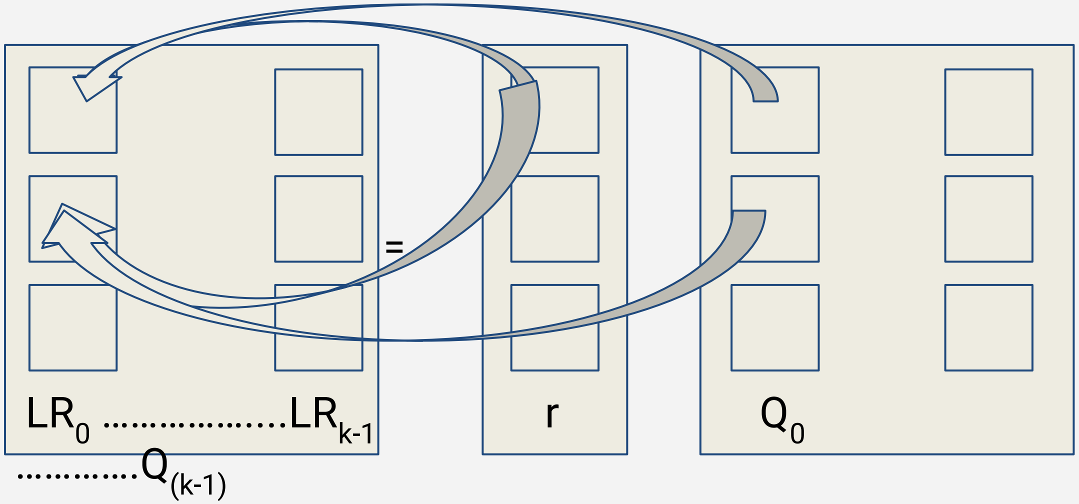
dotmat<<< grid(N,N),1 >>>

Each block calculates one entry of temp matrix with single threads

- Performs element wise multiplication

dot<<< 1,N >>> : Each threads add column of temp matrix to get array of k dot products

Step 2: Calculate matrix $(A_k.Q_n)Q_n$

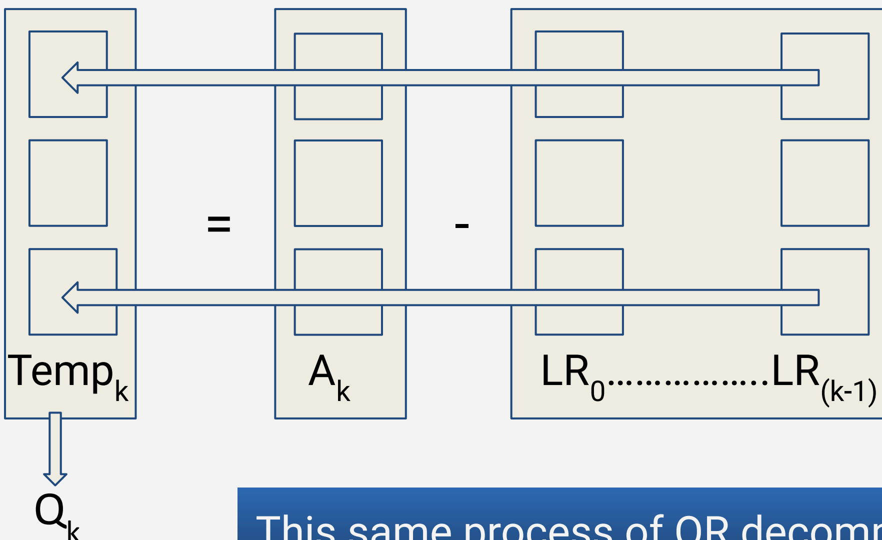


submat<<< grid(N,N),1 >>>

Each block calculates one entry of LR matrix with single threads

- Performs element wise multiplication and

Step 3: Calculate $Temp_k$ and Q_k



Qcal<<< 1,N >>>

- Each threads sums along rows of LR matrix and subtract the sum from corresponding element of A_k to get element of $Temp_k$
- Thread 0 calculated unit vector of $Temp_k$ i.e. Q_k

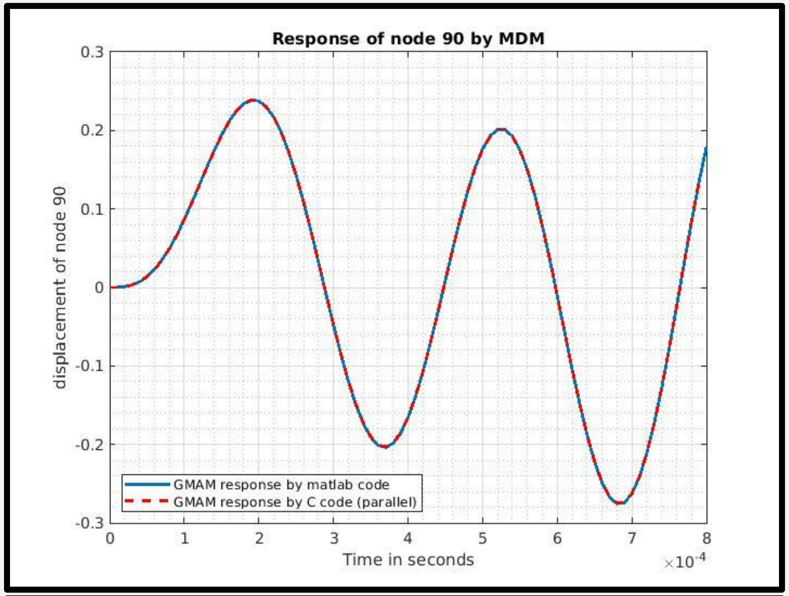
This same process of QR decomposition is done iteratively for n times to calculate whole Q matrix

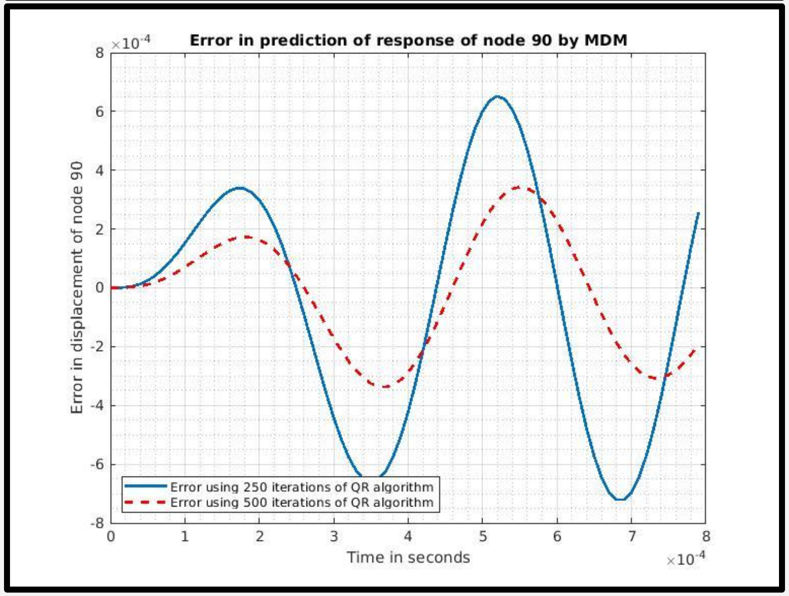
Results and discussions

Time study : The serial code took 48 mins to run and below is the time study for different techniques.

OpenMP (min)	OpenMPI (min)	CUDA (min)
8 thds - 9.1	8 PEs - 9.4	8 cores - 11.02
12 thds - 10.4	12 PEs - 9.3	12 cores - 10.9
16 thds - 12.5	16 PEs -10.7	16 cores - 9.7
20 thds - 15.03	20 PEs - 11.1	20 cores - 10.1

- The same problem was solved using Matlab code to confirm that the parallelized code are having data consistency and coherency
- The response of node 3781 (90 in reduced matrix) is plotted and its error value is also plotted. we can see that the error is of the order 10^{-4} (0.1%)





Work Distribution

1. Abhinandan Kumbhar - MPI, GPGPU programming, Solving same problem in MATLAB to compare results.
2. Jalaj Gupta - OpenMP, Solving same problem in MATLAB to compare results.
3. R. Nitin Iyer - MPI, Guyan condensation
4. Saurabh Pal - Guyan condensation, GPGPU programming

Everyone has contributed to Report writing, Slides preparation and Serial Code.