Report
Introduction to parallel scientific computing

Parallel QR Decomposition with Column Pivoting Using MPI

Submitted to

Dr Pawan kumar

Team numbers

2018201009 – Sarvat Ali
2018201046 – Divyanshi Kushwaha
2018201053 – Prabha Pandey

Work Distribution:

Research:

Done by all members

Sequential QR with column pivoting:

1) Making loaded matrix having both Q and R in it:
   Prabha Pandey
   Sarvat Ali
2) Extract Q from loaded matrix:
   Divyanshi Kushwaha

Parallel QR with column pivoting:

Done by all members

Description

QR decomposition of rank deficient matrix (rank(A)<number of columns(A)) cannot be calculated using traditional QR algorithm where we start iterating columns in order of their index, because this does not produces true orthonormal basis vectors. Hence QR decomposition with Column pivoting is used.
Parallelism to algorithm is applied using MPI (Message Passing Interphase).

Input

Matrix A with m rows and n columns.

Output

a). Modified matrix A that has Q and R
b). R matrix in its upper triangle
c). Vectors to calculate Q matrix (beta vector)
d). Array of coefficients which is used to recover Q from A.
e). Execution Time of both serial and parallel.

Language:

Python

MPI library:

mpi4py

Serial Implementation of algorithm is done using following steps.

A is matrix with m rows and n columns.

a). Norm of all columns(n) are calculated.

b).Swap the current column in A with column having maximum norm.

c). Householder vector of column whose norm is maximum is calculated as follows

$Q = I - \gamma u u^t$

where

u^t - transpose of vector u.

I - Identity matrix

γ = 2/square(norm(u)).

$u = [1\ x2/(\tau + x1)...xn/(\tau + x1)]^T$

τ = norm(x) where x is selected columns of matrix A.

d). This calculated Q is multiplied with matrix A and vector (u) is stored in A.

e). Submatrix of A is created leaving current colum and corresponding row.

These steps a,b,c,d,e are repeated on submatrix A until max-norm becomes becomes zero.

Here upper triangular of Matrix A is R matrix and Q is calculated using $Q = I - \gamma u u^t$ from vectors
u stored in A.

Parallel Implementation of algorithm using MPI.

A is matrix with m rows and n columns.

a). Process with rank = 0 recieves matrix A and splits matrix into blocks of size = m/number of processes. Where number of processes are excluding process with rank = 0

b). Each block of size (m/number of processes) is fed to different processors using MPI_Send and QR decomposition of each block is calculated using column pivoting as mentioned earlier.

c). Q and R matrices of each block from different processes is gathered inside process having rank=0.

d). QR decomposition of gathered R is performed using column pivoting as mentioned earlier.

e). From this QR decomposition we get final R matrix.

f). The resulting Q' is again divided into blocks and multiplied with older Q of each block to get final Q of matrix.

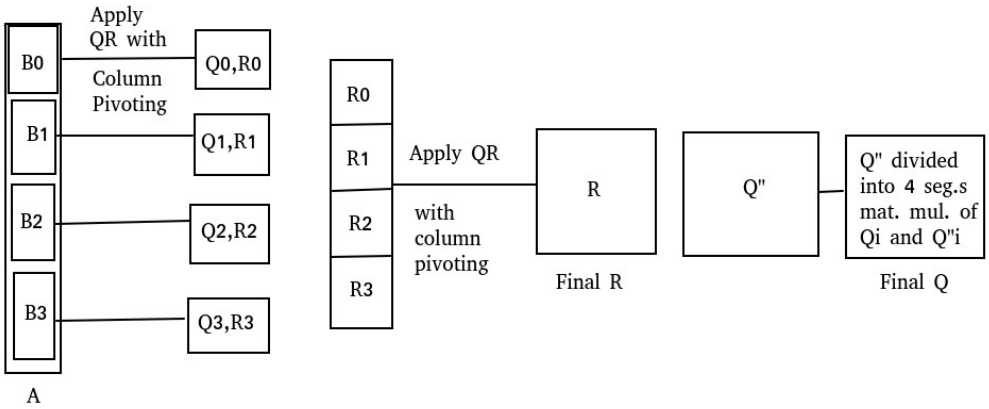
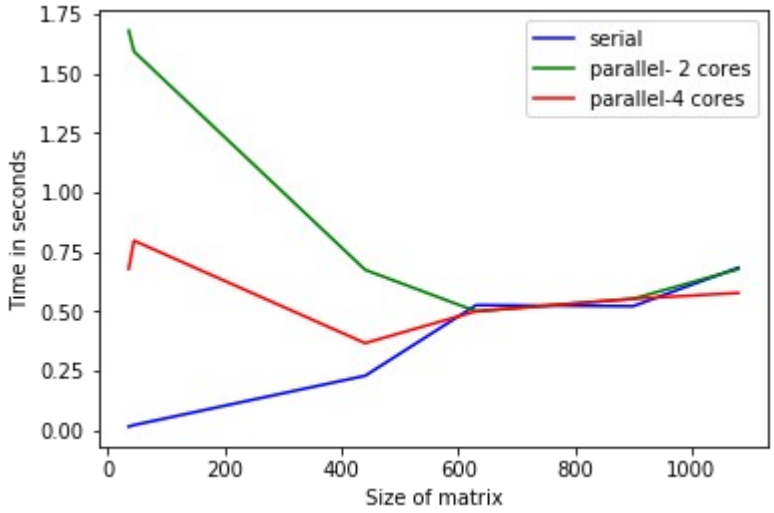


Figure to explain parallel QR approach.

Time comparisons

Size of matrix	Serial Execution Time	Parallel execution time
12*3(2 cores)	0.015413999557495117	1.681456722
12*3(4 cores)	0.015413999557495117	0.679453473997
15*3(2 cores)	0.022016048431396484	1.593206753
15*3(4 cores)	0.022016048431396484	0.797769302
21*21(2 cores)	0.22947797775268555	0.675009191997
21*21(4 cores)	0.22947797775268555	0.366607823998
30*21 (4 cores)	0.52649993896484375	0.500563796006
30*30(4 cores)	0.5219718933105469	0.553153249997
40*27(4 cores)	0.6514070034027096	0.5777665436

Graph Representing Comparison between Sequential and parallel QR decomposition



Refrences

http://web.mit.edu/ehliu/Public/sclark/Golub%20G.H.,%20Van%20Loan%20C.F.-%20Matrix%20Computations.pdf

http://honors.cs.umd.edu/uploads/thesis/file/177/TSQR_Final.pdf

https://mpi4py.readthedocs.io/en/stable/tutorial.html

https://web.stanford.edu/group/ctr/Summer/SP14/08_Transition_and_turbulence/08_sayadi.pdf