Parallelization of QR decomposition

Ondrej Zjevík

VSB-Technical University of Ostrava

5.12.2011

Contents

What is a QR decomposition?

It is

- $A = Q \cdot R$
- A is SPD (symmetric positive definite)
- Q is orthogonal
- R is upper triangular
- it is one way to compute eigen numbers

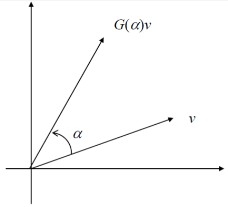
It is not

- the best way to compute solution of $Ax = B$
- the Gaussian elimination method is better

Basic facts

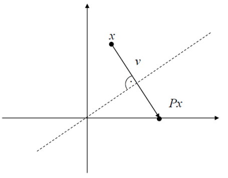
We make matrix A upper triangular by

Givens rotation



- numerically unstable
- requires one more cycle

Householder reflection



- numerically stable
- faster than Givens

Givens rotation

$$\begin{pmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{pmatrix} \cdot G_1 \rightarrow \begin{pmatrix} \star & \star & \star \\ \star & \star & \star \\ 0 & \star & \star \end{pmatrix} \cdot G_2 \rightarrow \begin{pmatrix} \star & \star & \star \\ 0 & \star & \star \\ 0 & \star & \star \end{pmatrix} \cdot G_3 \rightarrow \begin{pmatrix} \star & \star & \star \\ 0 & \star & \star \\ 0 & 0 & \star \end{pmatrix}$$

$$G(i,k,\alpha) = \begin{array}{cc} \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} & \begin{matrix} \\ \\ i \\ \\ k \\ \\ \\ \end{matrix} \end{array}$$

$$A \cdot (G_1 G_2 G_3) = R \rightarrow Q = G_3^T G_2^T G_1^T$$

Householder reflection

$$\begin{pmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{pmatrix} \cdot P_1 \rightarrow \begin{pmatrix} \star & \star & \star \\ 0 & \star & \star \\ 0 & \star & \star \end{pmatrix} \cdot P_2 \rightarrow \begin{pmatrix} \star & \star & \star \\ 0 & \star & \star \\ 0 & 0 & \star \end{pmatrix}$$

$$P_i = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \bigstar & \cdots & \bigstar \\ \vdots & \cdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \bigstar & \cdots & \bigstar \end{pmatrix}$$

$$A \cdot (P_1 P_2) = Q \rightarrow R = P_2^T P_1^T$$

Which one should we choose?

Which one should we choose?

Which one should we choose?

Which one should we choose?

Householder

How to use the QR decomposition to find eigenvalues?

It's simple. We compute matrices Q, R and update matrix A

$$A_{new} = Q' \cdot A \cdot Q.$$

We repeat QR decomposition with new matrix A. After 6 iterations for matrices less than 100 (approximately 10 iterations for matrices until 1000) the matrix A is like diagonal.

The items on diagonal of A are approximations of eigenvalues.

Example in matlab

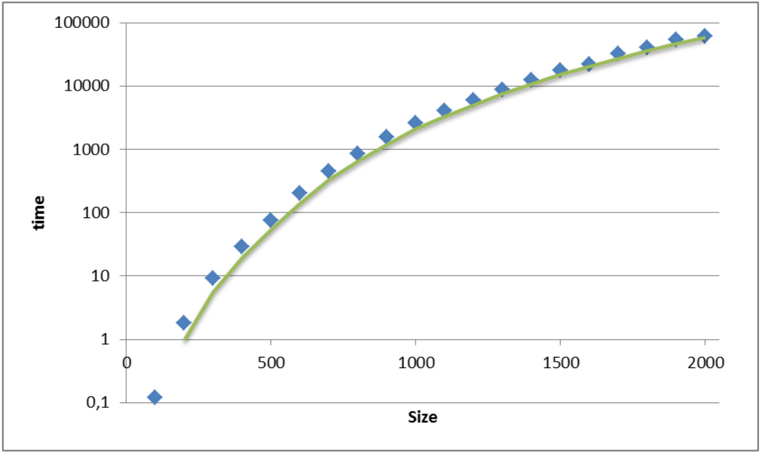
Givens rotation

```
[m,n] = size(A);
Q = eye(m,m); R = A;
for j=1:m
    for i = m:(-1):j+1
        x = R(:,j);
        if norm([x(i-1),x(i)]) > 0
            c = x(i-1)/norm([x(i-1),x(i)]);
            s=-x(i)/norm([x(i-1),x(i)]);
            G=eye(m);
            G([i-1,i],[i-1,i]) = [c,s;-s,c];
            R=G'*R;
            Q=Q*G;
        end
    end
end
```

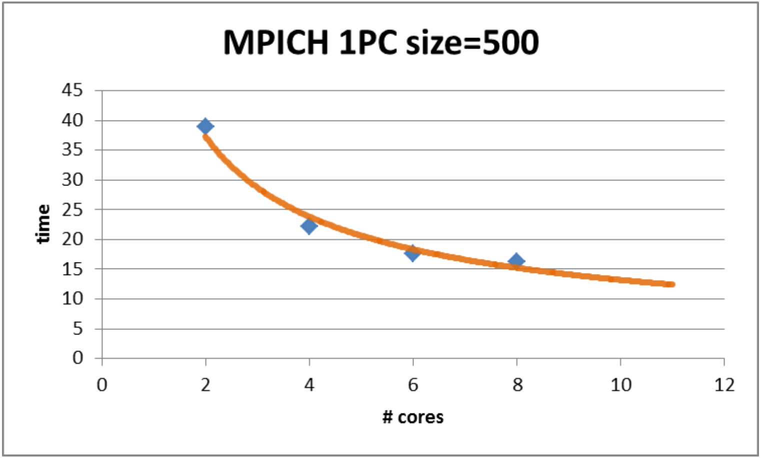
Hauseholder reflection

```
[m,n] = size(A);
Q = eye(m,m); R = A;
for j=1:min(m,n)
    x = R(j:m,j);
    v=sign(x(1))*norm(x)*eye(m-j+1,1)-x;
    if norm(v) > 0
        v=v/norm(v);
        P=eye(m);
        P(j:m,j:m)=P(j:m,j:m)-2*v*v';
        R=P*R;
        Q=Q*P;
    end
end
```

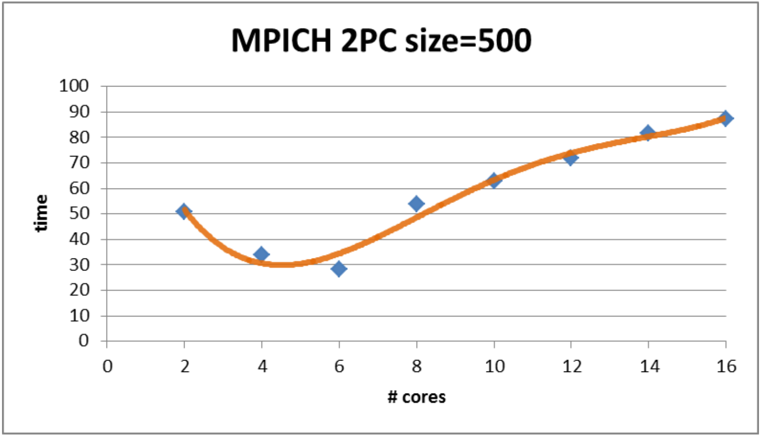
Sequential approach



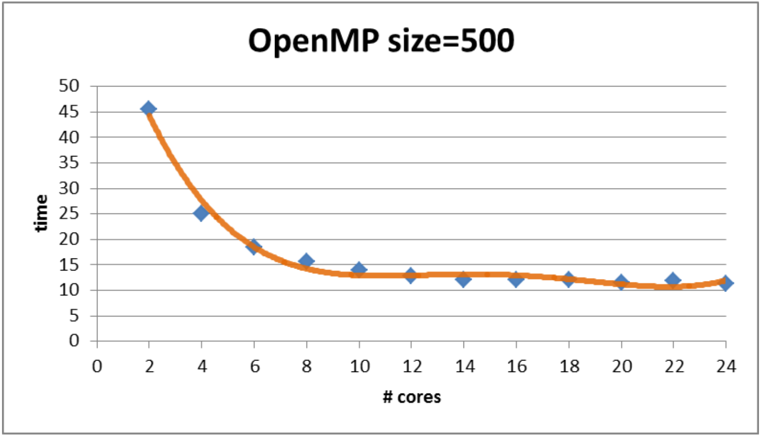
MPICH


MPICH 1PC size=500
time
45
40
35
30
25
20
15
10
5
0
0
2
4
6
8
10
12
cores

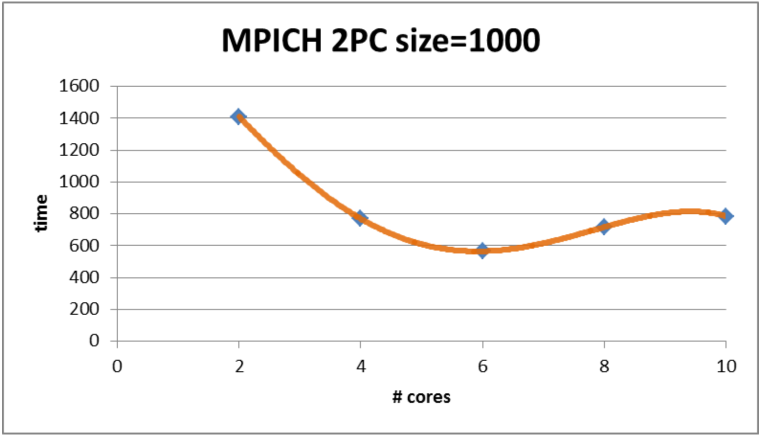
MPICH


MPICH 2PC size=500
time
100
90
80
70
60
50
40
30
20
10
0
0
2
4
6
8
10
12
14
16
cores

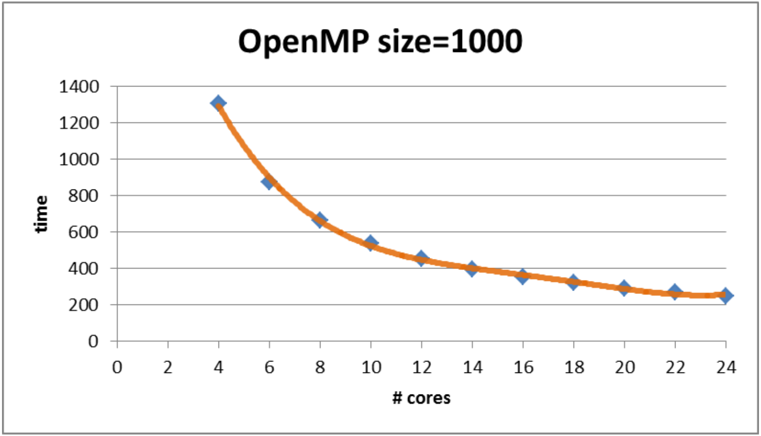
OpenMP


OpenMP size=500
time
cores
50
45
40
35
30
25
20
15
10
5
0
0
2
4
6
8
10
12
14
16
18
20
22
24

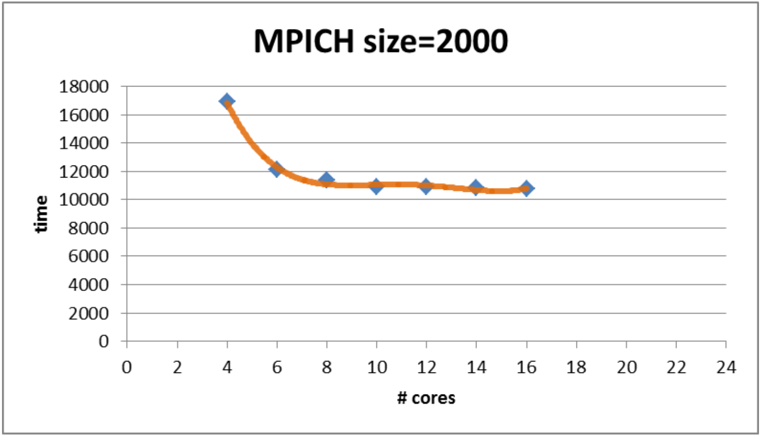
MPICH


MPICH 2PC size=1000
time
1600
1400
1200
1000
800
600
400
200
0
0
2
4
6
8
10
cores

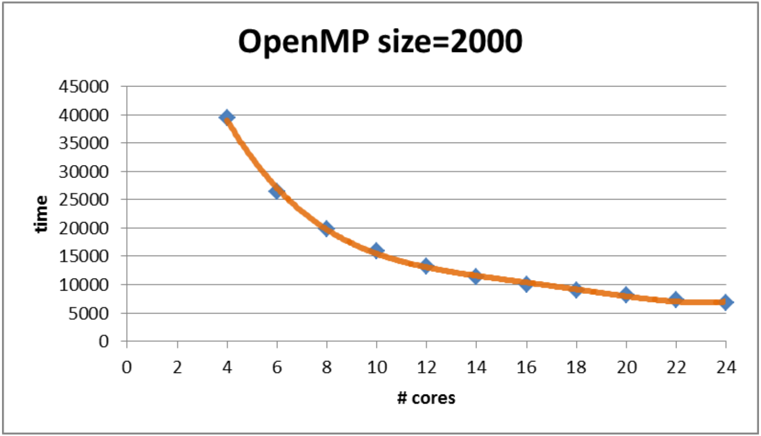
OpenMP


OpenMP size=1000
time
cores
1400
1200
1000
800
600
400
200
0
0 2 4 6 8 10 12 14 16 18 20 22 24

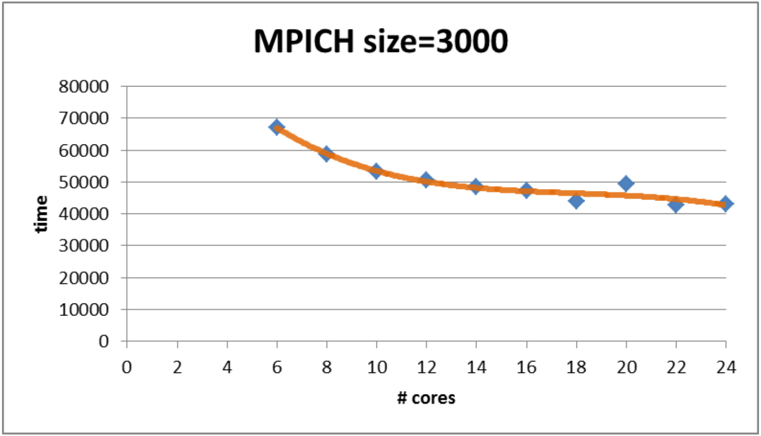
MPICH


MPICH size=2000
time
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
0
2
4
6
8
10
12
14
16
18
20
22
24
cores

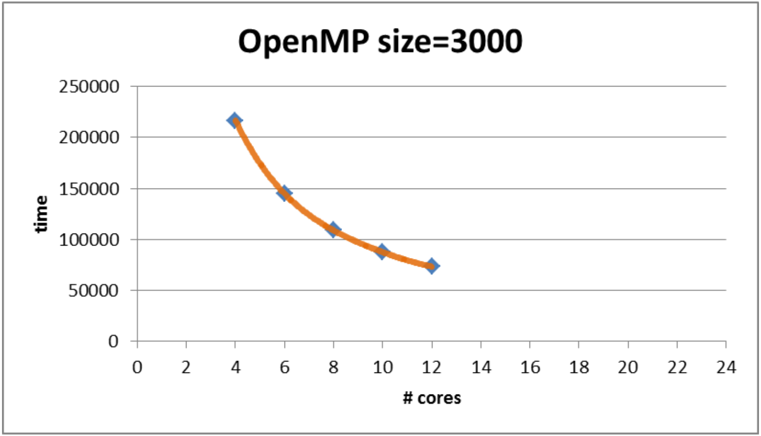
OpenMP


OpenMP size=2000
time
cores
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
0 2 4 6 8 10 12 14 16 18 20 22 24

MPICH


MPICH size=3000
time
80000
70000
60000
50000
40000
30000
20000
10000
0
0 2 4 6 8 10 12 14 16 18 20 22 24
cores

OpenMP


OpenMP size=3000
250000
200000
150000
100000
50000
0
time
0
2
4
6
8
10
12
14
16
18
20
22
24
cores

Thanks

Thank you!