Recursive least-squares using Householder transformations on massively parallel SIMD systems

E. J. Kontogiorghes\textsuperscript{a,c,1}, M. Clint\textsuperscript{b} and E. Dinenis\textsuperscript{c}

\textsuperscript{a} Department of Computer Science, Queen Mary and Westfield College, Mile End Road, London E1 4NS, UK
\textsuperscript{b} Dept. of Computer Science, The Queen's University of Belfast, Belfast, N. Ireland BT7 1NN, UK.
\textsuperscript{c} Centre for Mathematical Trading and Finance, City University Business School, Frobisher Crescent, Barbican Centre, London EC2Y 8HB, UK.

Abstract: Within the context of recursive least-squares, the implementation of a Householder algorithm for block updating the QR decomposition, on a massively parallel SIMD system, is considered. Initially, two implementations based on different mapping strategies for distributing the data matrices on the processing elements of the parallel computer are investigated. Timing models show that neither of these implementations is superior in all cases. In order to increase computational speed, a hybrid implementation uses performance models to partition the problem into two subproblems which are then solved by the first and second implementation, respectively.

Key words: Least-squares; Householder transformations; QR decomposition; Timing models; SIMD parallelism.

1 Introduction

It is frequently required to obtain updated least-squares solutions of a regression model where a block of data is repeatedly added. The updating can be computed by recursive formulae which have as a basic component the recalculation of the QR decomposition (QRD) [3,5,7,9,10,14]. The recursive least-squares (RLS) problem may be formulated as

\[
\min_{x_t} \| A_t x_t - y_t \|^2; \quad t = 1, 2, \ldots ,
\]

where \( \| \cdot \| \) denotes Euclidean length, \( A_t \) is an \( m_t \times (n - 1) \) full column rank matrix \( (m_t \geq n) \), \( y_t \in \mathbb{R}^n \) is the response vector, \( x_t \) is the unknown vector of

\textsuperscript{1} Corresponding author. Email: ricos@dcs.qmw.ac.uk
\( n - 1 \) coefficients and
\[
\hat{A}_t = \begin{pmatrix} A_t & y_t \end{pmatrix} = \begin{pmatrix} \hat{A}_{t-1} \\ \hat{m}_t \end{pmatrix} \begin{pmatrix} m_{t-1} \\ \hat{m}_t \end{pmatrix} \quad (m_0 = 0).
\] (2)

Given the QRD
\[
Q^T_t \hat{A}_t = \begin{pmatrix} \hat{R}_t \\ 0 \end{pmatrix} = \begin{pmatrix} R_t & u_t \\ 0 & s_t \end{pmatrix} \begin{pmatrix} n - 1 \\ 1 \end{pmatrix}
\] (3)

the LS estimator of \( x_t \) is derived from the solution of \( R_t x_t = u_t \), where \( R_t \) is an upper triangular non-singular matrix and \( Q_t \) is an \( m_t \times m_t \) orthogonal matrix. By computing the orthogonal factorization
\[
\tilde{Q}^T_{t+1} \hat{A}_{t+1} = \begin{pmatrix} \tilde{R}_{t+1} \\ 0 \end{pmatrix} = \begin{pmatrix} R_{t+1} & u_{t+1} \\ 0 & s_{t+1} \end{pmatrix}
\] (4)

the updated LS estimator of \( x_{t+1} \) may be obtained by solving
\[
R_{t+1} x_{t+1} = u_{t+1},
\] (5)

where \( \tilde{Q}_{t+1} \) is an \( (n + \tilde{m}_{t+1}) \times (n + \tilde{m}_{t+1}) \) orthogonal matrix and \( \tilde{R}_{t+1} \in \mathbb{R}^{n \times n} \) is upper triangular. Thus, after computing the QRD of \( \hat{A}_1 \), the recursive formulae (4) and (5) may be used to derive the updated LS estimator of \( x_{t+1} \) for \( t = 1, 2, \ldots \). Observe that the orthogonal matrix \( Q^T_t \) in the QRD of \( \hat{A}_{t+1} \) is given by
\[
\begin{pmatrix} \tilde{Q}^T_{t+1} \\ 0 \end{pmatrix} = \begin{pmatrix} I_n & 0 \\ 0 & I_{\tilde{m}_{t+1}} \end{pmatrix} \begin{pmatrix} \tilde{R}^T_t \\ 0 \end{pmatrix} \begin{pmatrix} I_n & 0 \\ 0 & I_{\tilde{m}_{t+1}} \end{pmatrix}
\]

Householder reflections and Givens rotations are the main numerically stable methods used to compute the factorization in (4) [4–10,16,17,19]. Givens method is found to be superior in speed to Householder method only when the blocks of new observations \( \hat{A}_t \) comprise a small number of rows compared with the number of regressors \([7,10]\). Under the assumption that \( \tilde{m}_{t+1} > n \), parallel algorithms based on Householder transformations are considered below for computing the factorization (4) on a massively parallel SIMD machine. The orthogonal matrix \( Q_{t+1} \) will not be explicitly constructed and the trivial solution of the triangular system (5) will not be discussed.

2 SIMD Implementation

The orthogonal matrix \( Q^T_t \) in (4) may be defined as a product of Householder reflectors, where the \( i \)th reflector reduces to zero the \( i \)th column of \( \hat{A}_{t+1} \) by preserving the upper triangular structure of \( \hat{R}_t \). Figure 1 shows the steps in applying the Householder transformations in a data-parallel mode, where \( \hat{R}_t \) is overwritten by \( \tilde{R}_{t+1} \). For notational convenience we let \( m = \tilde{m}_{t+1} \) and denote the matrices \( \hat{R}_t \) and \( \hat{A}_{t+1} \) by \( R \) and \( A \), respectively.
The Householder algorithm has been implemented in MasPar-Fortran on the MasPar MP-1208 [15]. This SIMD system has 8192 Processing Elements (PEs) arranged in a 2D-array of size $e_2 \times e_1$, where $e_1 = 128$ and $e_2 = 64$. The main mapping layouts for distributing the $m \times n$ matrix $A$ over the PEs are the (default) cyclic, column and row layouts, which use $[m/e_1][n/e_2]$, $n[m/e_1 e_2]$ and $m[n/e_1 e_2]$ layers of memory, respectively [1]. The memory layers have dimension $e_2 \times e_1$ and are assumed to be arranged in an $M \times N$ grid $G$, where $M$ and $N$ depend on the chosen layout. Under the cyclic-layout, the $i$th row of $A$, $A_{:,i}$, resides in the $j$th column of layers $G_{k,j}$, where $j = ((i - 1) \mod e_1) + 1$, $k = [i/e_1]$, $M = [m/e_1]$ and $N = [n/e_1]$. In the column-layout the $i$th column of $A$, $A_{i,:}$, resides in $G_{:,i}$, where $M = [m/e_1 e_2]$ and $N = n$, while in the row-layout $A_{:,i}$ resides in $G_{i,:}$ where $M = m$ and $N = [n/e_1 e_2]$.

A mapping layout is chosen so that the maximum number of PEs remains active during the computations without, however, increasing the communication overheads between the PEs. Since $m > n$, the row-layout will be inefficient compared with the column-layout. Consequently, the performances of the Householder algorithm using cyclic and column layouts are considered. The computational details of the implementations on the MasPar are not shown. However, a pseudo code version is given in Fig. 1. Similar implementations have been previously considered for computing the QRD under the assumption that the dimensions of the matrices are exact multiples of the corresponding dimensions of the array processor and that none of their dimensions exceed $e_1 e_2$ [2,11,12]. Here the only constraint imposed on the dimensions of the matrices is that $m > n$.

```
for $i = 1, 2, \ldots, n$ do
  $s := \sqrt{R_{i,i}^2 + ||A_{:,i}||^2}$
  if ($R_{i,i} < 0$) then $s := -s$
  $\eta := R_{i,i} + s$
  $c := s \times \eta$
  $Z^T := (\eta \times R_{i,i} + A_{:,i}^T A_{:,i}) / c$
  $R_{i,i} := R_{i,i} - \eta \times Z^T$
  $A_{:,i} := A_{:,i} - A_{:,i} Z^T$
end-do
```

Fig. 1. The pseudo data-parallel Householder algorithm.

2.1 Performance Models

Statistical methods are employed to construct performance models for the Householder implementations. These models can realistically evaluate the execution speed of the algorithms for various values of $m$ and $n$. The order of
complexity of the algorithms will be the same if other 2D SIMD array processors are used.

Using a cyclic-layout to map the matrices \( A \in \mathbb{R}^{m \times n} \) and \( R \in \mathbb{R}^{n \times n} \) onto the PE array, the time (sec \( \times 10^{-3} \)) required to apply the \( i \)th Householder transformation is found to be

\[
\Phi_i(m, n, i) = 10.65\left[(n + 1 - i) / e_2\right] + 1.66\left[(n + 1 - i) / e_2\right][m / e_1] \\
+ 4.72[m / e_1] + 0.03(n + 1 - i)[m / e_1].
\]

(6)

The explanatory factors of the above timing model have been derived by considering the number of layers involved in the arithmetic computations and the number of times layers are replicated or reduced. Regression analysis is then employed to find the least squares estimators of the coefficients in the model.

The total time spent in applying the Householder reflections is thus given by \( \sum_{i=1}^{n} \Phi_i(m, n, i) \), which is equivalent to:

\[
\Phi_2(m, n) = e_2 \sum_{i=1}^{N} (10.65i + 1.66Mi + 4.72M) + 0.03(n + 1)M/2 \\
- (N e_2 - n)(10.65N + 1.66NM + 4.72M),
\]

(7)

where \( N = [n / e_2] \) and \( M = [m / e_1] \). However, the overheads of the implementation which are mainly the passing of arguments (subarrays) to various routines are not included in this model. Evaluating \( \Phi_2(m, n) \), and using backward stepwise regression on a sample of more than 5000 execution times, a highly accurate timing model for the cyclic-layout Householder implementation is found to be:

\[
T_1(m, n) = N(1391.0 + 66.8N^2 + 388.2M + 115.5NM) \\
- (N e_2 - n)(17.42 + 4.53M + 2.79N^2 + 3.66NM) \\
= N(276.12 + 98.28M - 118.74NM - 111.76N^2) \\
+ n(17.42 + 4.53M + 2.79N^2 + 3.66NM).
\]

(8)

Calculations shown that the residuals are normally distributed. Thus, the hypothesis tests made during the selection of this model are justified [18]. The adequacy of the latter model, measured by the coefficient of determination, is found to be 99.99%. Figure 2 shows the ratio between the predicted and actual execution times using \( T_1(m, n) \) and \( \Phi_2(m, n) \). It may be observed that, overall, the predictions given by \( T_1(m, n) \) are more accurate than those of \( \Phi_2(m, n) \).

Using cyclic and column layouts to map respectively the matrices \( R \) and \( A \) onto the PEs, a model for estimating the execution time of the \( i \)th Householder reflection is:

\[
\Phi_3(m, n, i) = c_0 + c_1(n + 1 - i) + c_2(n + 1 - i)[m / e_1 e_2] + c_3[n / e_2],
\]

(9)

where \( c_0, \ldots, c_3 \) are constants. Evaluating \( \sum_{i=1}^{n} \Phi_3(m, n, i) \) and using regression analysis, the execution time of the column-layout implementation is found
Fig. 2. Ratio between predicted and actual execution times.

to be:

\[ T_2(m, n) = n \left( 13.51 + 2.69n + 1.33(n + 1)\left[ \frac{m}{e_1 e_2} \right] + 2.98\left[ \frac{n}{e_2} \right] \right). \quad (10) \]

From Fig. 3 it can be observed that neither of the implementations is superior in all cases. The efficiency of the cyclic-layout implementation improves compared with that of the column-layout implementation, for fixed \( m \) and increasing \( n \). Table 1 shows that the column-layout is superior for very large \( m \) and relatively small \( n \).

Fig. 3. Ratio of the execution times produce by the models of the cyclic-layout and column-layout implementations.

3 The Hybrid SIMD Implementation

The results above suggest that the application of the \( n \) required Householder reflections be divided into two parts. In the first part, \( n_1 \) reflections are applied
to annihilate the first \( n_1 \) columns of \( A \) using cyclic-layout; in the second stage the remaining \( n_2 = n - n_1 \) reflections reduce to zero the submatrix \( A_{n_1+1} \) using column-layout, where \( A_{n_1+1} \) comprise the last \( n_2 \) columns of \( A \). Let \( \tilde{T}_1(m, n, n_1) \) be the time required to complete the first stage. Then the total execution time of the hybrid implementation is given by

\[
T_4(m, n, n_1) = \tilde{T}_1(m, n, n_1) + T_2(m, n - n_1) + T_3(m, n - n_1),
\]

where

\[
T_3(m, n - n_1) = \left[ m/e_1 \right] (65.45 \left[ (n - n_1)/e_2 \right] + 0.62(n - n_1))
\]

is the time (sec \( \times 10^{-6} \)) required to remap the submatrix \( A_{n_1+1} \) from cyclic-layout to column-layout.

The value of \( n_1 \), which may not be unique, is chosen to minimize \( T_4(m, n, n_1) \). That is, \( n_1 \) is the solution of

\[
\arg\min_{n_1} T_4(m, n, n_1) \quad \text{subject to} \quad \left\{ \begin{array}{l}
0 \leq n_1 \leq n \\
n_1 \text{ is integer}
\end{array} \right.,
\]

which may be easily determined by (simultaneously) computing \( T_4(m, n, n_1) \) for \( n_1 = 0, \ldots, n \) and selecting the value(s) which minimize \( T_4 \). Note that \( T_4(m, n, n_1) \) is a statistical timing model and includes an unpredicted random error component. Consequently, the solution of (12) may not yield the true value of \( n_1 \) which minimizes the execution time of the hybrid algorithm.

An alternative method for estimating \( n_1 \) is to compare the total number of memory layers used for different values of \( n_1 \). In this case \( n_1 \) is the solution of

\[
\arg\min_{n_1} \left( \sum_{i=1}^{n_1} \left[ m/e_1 \right] (n+1-i)/e_2 + \frac{1}{2}(n-n_1)(n-n_1+1) \frac{m}{e_1 e_2} \right),
\]

where \( 0 \geq n_1 \geq n \). This estimation method does not, however, take into account the cost of remapping \( A_{n_1+1} \). Better estimates of \( n_1 \) might possibly be obtained by constructing more accurate timing models than that given by (12), or by introducing a weighting factor into (13) which will take account of the implementation overheads.

Table 2 shows the actual execution times for the three implementations, where the negligible time required to compute the estimates of \( n_1 \) is not included and \( \tilde{T}_1(m, n, n_1) \) has been computed as \( \sum_{i=1}^{n_2} \Phi_i(m, n, i) \). The estimates for \( n_1 \) obtained using (12) and (13) are denoted by \( \hat{n}_1 \) and \( n_1^* \), respectively. In most cases, the two estimation methods yield different values for \( n_1 \). The execution time for the hybrid implementation, using the estimation \( n_1^* \), is found to be more accurate than those using \( \hat{n}_1 \) in approximately half of the cases. In some instances, for very small \( n \) and relatively large \( M \), the hybrid implementation
<table>
<thead>
<tr>
<th>M</th>
<th>n</th>
<th>(a_1)</th>
<th>Hybrid (n_1^*)</th>
<th>Hybrid Cyclic Column with (a_1)</th>
<th>M</th>
<th>n</th>
<th>(a_1)</th>
<th>Hybrid (n_1^*)</th>
<th>Hybrid Cyclic Column with (a_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>32</td>
<td>25</td>
<td>3.52</td>
<td>23</td>
<td>2.81</td>
<td>3.28</td>
<td>3.16</td>
<td>10</td>
<td>128</td>
</tr>
<tr>
<td>10</td>
<td>64</td>
<td>57</td>
<td>6.10</td>
<td>55</td>
<td>6.09</td>
<td>6.33</td>
<td>18.05</td>
<td>10</td>
<td>160</td>
</tr>
<tr>
<td>10</td>
<td>96</td>
<td>89</td>
<td>10.78</td>
<td>87</td>
<td>11.01</td>
<td>10.78</td>
<td>39.61</td>
<td>10</td>
<td>192</td>
</tr>
<tr>
<td>20</td>
<td>32</td>
<td>16</td>
<td>4.21</td>
<td>13</td>
<td>4.69</td>
<td>4.82</td>
<td>5.68</td>
<td>20.68</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>64</td>
<td>48</td>
<td>10.78</td>
<td>45</td>
<td>10.54</td>
<td>11.24</td>
<td>19.46</td>
<td>20</td>
<td>160</td>
</tr>
<tr>
<td>20</td>
<td>96</td>
<td>88</td>
<td>10.98</td>
<td>77</td>
<td>18.99</td>
<td>19.45</td>
<td>41.24</td>
<td>20</td>
<td>192</td>
</tr>
<tr>
<td>30</td>
<td>32</td>
<td>7</td>
<td>5.63</td>
<td>3</td>
<td>5.85</td>
<td>7.97</td>
<td>9.35</td>
<td>30</td>
<td>128</td>
</tr>
<tr>
<td>30</td>
<td>64</td>
<td>28</td>
<td>11.20</td>
<td>26</td>
<td>11.76</td>
<td>12.02</td>
<td>15.38</td>
<td>30</td>
<td>160</td>
</tr>
<tr>
<td>30</td>
<td>96</td>
<td>57</td>
<td>26.49</td>
<td>57</td>
<td>26.01</td>
<td>28.13</td>
<td>32.89</td>
<td>30</td>
<td>192</td>
</tr>
<tr>
<td>50</td>
<td>32</td>
<td>103</td>
<td>37.97</td>
<td>99</td>
<td>37.97</td>
<td>39.61</td>
<td>72.99</td>
<td>50</td>
<td>128</td>
</tr>
<tr>
<td>50</td>
<td>64</td>
<td>20</td>
<td>6.80</td>
<td>20</td>
<td>6.60</td>
<td>10.94</td>
<td>6.79</td>
<td>50</td>
<td>160</td>
</tr>
<tr>
<td>50</td>
<td>96</td>
<td>31</td>
<td>28.27</td>
<td>31</td>
<td>27.31</td>
<td>37.97</td>
<td>44.29</td>
<td>50</td>
<td>192</td>
</tr>
<tr>
<td>100</td>
<td>32</td>
<td>31</td>
<td>47.81</td>
<td>31</td>
<td>47.82</td>
<td>52.04</td>
<td>75.24</td>
<td>100</td>
<td>128</td>
</tr>
<tr>
<td>100</td>
<td>64</td>
<td>50</td>
<td>38.67</td>
<td>0</td>
<td>46.18</td>
<td>45.47</td>
<td>45.94</td>
<td>100</td>
<td>160</td>
</tr>
<tr>
<td>100</td>
<td>96</td>
<td>50</td>
<td>38.67</td>
<td>0</td>
<td>46.18</td>
<td>45.47</td>
<td>45.94</td>
<td>100</td>
<td>192</td>
</tr>
<tr>
<td>100</td>
<td>128</td>
<td>62</td>
<td>57.43</td>
<td>0</td>
<td>65.65</td>
<td>63.98</td>
<td>77.58</td>
<td>100</td>
<td>224</td>
</tr>
<tr>
<td>100</td>
<td>256</td>
<td>62</td>
<td>37.43</td>
<td>0</td>
<td>45.65</td>
<td>44.98</td>
<td>45.47</td>
<td>100</td>
<td>256</td>
</tr>
<tr>
<td>100</td>
<td>512</td>
<td>62</td>
<td>37.43</td>
<td>0</td>
<td>45.65</td>
<td>44.98</td>
<td>45.47</td>
<td>100</td>
<td>256</td>
</tr>
<tr>
<td>200</td>
<td>32</td>
<td>31</td>
<td>47.81</td>
<td>31</td>
<td>47.82</td>
<td>52.04</td>
<td>75.24</td>
<td>200</td>
<td>128</td>
</tr>
<tr>
<td>200</td>
<td>64</td>
<td>50</td>
<td>38.67</td>
<td>0</td>
<td>46.18</td>
<td>45.47</td>
<td>45.94</td>
<td>200</td>
<td>160</td>
</tr>
<tr>
<td>200</td>
<td>96</td>
<td>50</td>
<td>38.67</td>
<td>0</td>
<td>46.18</td>
<td>45.47</td>
<td>45.94</td>
<td>200</td>
<td>192</td>
</tr>
<tr>
<td>200</td>
<td>128</td>
<td>62</td>
<td>57.43</td>
<td>0</td>
<td>65.65</td>
<td>63.98</td>
<td>77.58</td>
<td>200</td>
<td>224</td>
</tr>
<tr>
<td>200</td>
<td>256</td>
<td>62</td>
<td>37.43</td>
<td>0</td>
<td>45.65</td>
<td>44.98</td>
<td>45.47</td>
<td>200</td>
<td>256</td>
</tr>
<tr>
<td>200</td>
<td>512</td>
<td>62</td>
<td>37.43</td>
<td>0</td>
<td>45.65</td>
<td>44.98</td>
<td>45.47</td>
<td>200</td>
<td>256</td>
</tr>
</tbody>
</table>

is reduced to the column-layout implementation, that is, the estimated value of \(n_1\) is zero (see for example the case \(n = 32\) and \(M = 100\)). With the exception of few cases, the hybrid implementation is more efficient than both the cyclic-layout and column-layout implementations.

### 4 Conclusion

The performance, on a SIMD computer, of the data-parallel Householder algorithm for computing the orthogonal factorization given by (4), has been considered. Timing models have been constructed for estimating the execution speed of the algorithm when cyclic-layout and column-layout mapping strategies are used. These accurate timing models revealed that neither of the data-mapping distributions is superior for all values of \(m\) and \(n\) \((m > n)\). A new hybrid implementation which switches from cyclic-layout to column-
layout has been proposed. The hybrid algorithm first applies $n_1$ ($0 \geq n_1 \geq n$) transformations under the cyclic-layout regime, and then the column-layout implementation is employed to complete the factorization.

Two methods have been used for deriving an estimate for $n_1$. The first method is based on minimizing the estimated execution time provided by the timing model for the algorithm, while the second method is based on minimizing the total number of memory layers used. The two estimators of $n_1$ are not always identical and are probably different from the optimum value of $n_1$, which minimizes the execution time of the hybrid algorithm. However, in nearly all the experiments performed, the hybrid algorithm using both estimators of $n_1$ is found to have the best performance of the three candidates. The improvement in performance become significant for real-time applications in which a very large number of data updateings are required.

For other SIMD systems, the value of $n_1$ may be best derived using the straightforward minimization of the total number of memory layers used, rather than minimizing the estimated time given by performance model in (11) which requires the time consuming re-determination of the coefficients of the various timing models.

Similar hybrid algorithms may be used to compute the orthogonal factorization (4), based on Givens rotations and Householder reflections, when $m < n$. In this case, the efficiency of the row-layout mapping strategy should be investigated as a (possibly) more efficient alternative. The hybrid approach could also be employed to improve the efficiency of the SIMD algorithms proposed in [3,9–13].

Acknowledgements

The work was carried out using the facilities of the London Parallel Applications Centre at Queen Mary and Westfield College, University of London. The research of the first author was supported by the C.T. Bowring Company Ltd. The authors are grateful to Prof. Dennis Parkinson for his constructive comments.

References


1985.


factorizations: parallel implications. *Journal of Computational and Applied


University of Kent, UK, 1991.