The Technique ”Evaluation-Interpolation” in Parallel Processing with Matlab

Dimitris Varsamis

Department of Informatics Engineering,
Technological Educational Institute of Central Macedonia - Serres
62124, Serres, Greece

Copyright © 2017 Dimitris Varsamis. This article is distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

In this paper a new parallel algorithm for the computation of the inverse of a bivariate polynomial matrix are presented. The parallel algorithm based on the technique evaluation-interpolation and for the part of interpolation uses the Newton bivariate polynomial interpolation. The algorithm is applied to the programming environment of MATLAB with Parallel Computing Toolbox and is compared to the corresponding build-in function of MATLAB inv().

Keywords: Parallel Algorithm, Bivariate polynomial interpolation, Evaluation-Interpolation, Matlab PCT

1 Introduction

The problem of computing the inverse of a polynomial matrix has been considered by many authors due to its large number of applications i.e. calculation of the transfer function matrix of a system [2], solution of Auto-Regressive equations [1], coding and cryptography [5] etc. The inverse of a polynomial matrix can be computed either by using symbolic algorithms, like the Leverrier-Faddev algorithm [4], or numerical algorithms [3, 10, 6, 9]. Among those algorithms it is shown that DFT interpolation techniques are the most promising as concerns the running time, in contrast to the symbolic ones which are accurate but time consuming. In order to compute the inverse of a polynomial matrix,
interpolation algorithms initially compute the values of the determinant and
the adjoint of the polynomial matrix at specific interpolation points, and then
use known interpolation techniques in order to reconstruct the determinantal
polynomial and the adjoint polynomial matrix.

In this paper, we start by presenting in Section 2, the computation of the
inverse of a two-variable polynomial matrix base to algorithm in [6] the optimal
base in [7] and optimal degree in [8]. In section 3, the parallel approach is
presented. Also, the execution times between sequential algorithm, parallel
algorithm and the corresponding build-in function of MATLAB inv() is given.

2 On the computation of the inverse of a two-variable polynomial matrix

The calculation of the inverse of a two variable polynomial matrix are imple-
mented with the ”Evaluation-Interpolation” technique as follow:

Algorithm 1: Let \( A(x, y) \in \mathbb{R}[x, y]^{m \times m} \) a two variable polynomial matrix

\[
A(x, y) = \begin{pmatrix}
a_{1,1}(x, y) & \cdots & a_{1,m}(x, y) \\
\vdots & \ddots & \vdots \\
a_{m,1}(x, y) & \cdots & a_{m,m}(x, y)
\end{pmatrix}
\]

Step 1: Computation of the degree of the determinant and the degree of each
element of the adjoint matrix.
We find the degree matrices \( D = [d_{i,j}], D^x = [d^x_{i,j}] \) and \( D^y = [d^y_{i,j}] \) where
\( d_{i,j} = \deg_{x,y}\{a_{i,j}(x, y)\} \), is the total degree of the polynomial \( a_{i,j}(x, y) \), \( d^x_{i,j} = \deg_x\{a_{i,j}(x, y)\} \), is the degree respect to \( x \) of the polynomial \( a_{i,j}(x, y) \) and
\( d^y_{i,j} = \deg_y\{a_{i,j}(x, y)\} \), is the degree respect to \( y \) of the polynomial \( a_{i,j}(x, y) \) as
defined in [8].
For the determinant we have

\[
d_{\text{max}} = \text{DegDet}(D), \quad d^x_{\text{max}} = \text{DegDet}(D^x), \quad d^y_{\text{max}} = \text{DegDet}(D^y)
\]

The \( \text{DegDet}(D_{1,1}) \) is defined in [8] and given by

\[
\text{DegDet}(D) = \begin{cases}
\max_{1 \leq j \leq m} \{d_{i,j} + \text{DetDeg}(D_{i,j})\} & m > 2, \ i \ \text{constant} \\
\max \{d_{1,1} + d_{2,2}, d_{1,2} + d_{2,1}\} & m = 2
\end{cases}
\]

Then, for the adjoint we define the matrix \( \bar{D} \in \mathbb{N}^{m \times m} \) which is given by

\[
\bar{D} = (\bar{d}_{i,j}) = \begin{pmatrix}
\text{DegDet}(D_{1,1}) & \cdots & \text{DegDet}(D_{1,m}) \\
\vdots & \ddots & \vdots \\
\text{DegDet}(D_{m,1}) & \cdots & \text{DegDet}(D_{m,m})
\end{pmatrix}
\]
Technique "evaluation-interpolation" 1795

where $D_{i,j} \in \mathbb{N}^{(m-1)\times(m-1)}$ submatrix of matrix $D$ which the $i$ row and the $j$ column are deleted.

Accordingly, we define the matrix $\bar{D}^x \in \mathbb{N}^{m\times m}$ which is given by $\bar{D}^x = (\bar{d}^x_{i,j}) = (\text{DegDet} (D^x_{i,j}))$ where $D^x_{i,j} \in \mathbb{N}^{(m-1)\times(m-1)}$ submatrix of matrix $D^x$ which the $i$ row and the $j$ column are deleted and the matrix $\bar{D}^y \in \mathbb{N}^{m\times m}$ which is given by $\bar{D}^y = (\bar{d}^y_{i,j}) = (\text{DegDet} (D^y_{i,j}))$ where $D^y_{i,j} \in \mathbb{N}^{(m-1)\times(m-1)}$ submatrix of matrix $D^y$ which the $i$ row and the $j$ column are deleted.

**Step 2:** Computation of the number of required points. This computation are presented in [8].

For the determinant the optimal number of required interpolation points is given by

$$M_o = \begin{cases} 
(d_{max}+2) / d_{max} & \text{for } d^x_{max} + d^y_{max} = 2 \cdot d_{max} \\
(d^x_{max} + 1) / (d^y_{max} + 1) & \text{for } d^x_{max} + d^y_{max} = d_{max} \\
(d_{max}+2) / d_{max} - (S_{d_{max}} - d^x_{max} + S_{d_{max}} - d^y_{max}) & \text{for } d_{max} < d^x_{max} + d^y_{max} < 2 \cdot d_{max}
\end{cases}$$

Then, the maximum element of matrix $\bar{D}$ is the greatest degree of polynomial elements of the adjoint matrix $\text{adj}A(x, y)$ and given by

$$\bar{d}_{max} = \max_{1 \leq i,j \leq m} \{\bar{d}_{i,j}\}$$

The maximum element of matrix $\bar{D}^x$ is the greatest degree of polynomial elements of the adjoint matrix $\text{adj}A(x, y)$ respect to $x$ and given by

$$\bar{d}^x_{max} = \max_{1 \leq i,j \leq m} \{\bar{d}^x_{i,j}\}$$

The maximum element of matrix $\bar{D}^y$ is the greatest degree of polynomial elements of the adjoint matrix $\text{adj}A(x, y)$ respect to $y$ and given by

$$\bar{d}^y_{max} = \max_{1 \leq i,j \leq m} \{\bar{d}^y_{i,j}\}$$

Thus, the optimal number of required interpolation points is given by

$$N_o = \begin{cases} 
(d_{max}+2) / d_{max} & \bar{d}^x_{max} + \bar{d}^y_{max} = 2 \cdot \bar{d}_{max} \\
(d^x_{max} + 1) / (d^y_{max} + 1) & \bar{d}^x_{max} + \bar{d}^y_{max} = \bar{d}_{max} \\
(d_{max}+2) / d_{max} - (S_{d_{max}} - \bar{d}^x_{max} + S_{d_{max}} - \bar{d}^y_{max}) & \bar{d}_{max} < \bar{d}^x_{max} + \bar{d}^y_{max} < 2 \cdot \bar{d}_{max}
\end{cases}$$

In the first case of $M_o$ or $N_o$ are used the triangular basis, in the second case of $M_o$ or $N_o$ are used the orthogonal basis and in the third case of $M_o$ or $N_o$ are
used the polygonal basis.

**Step 3:** Evaluation part

We define the set of required points on triangular basis as $S_{\Delta}^{(n)}$ or on orthogonal basis as $\tilde{S}_{\Delta}^{(k_1,k_2)}$ or on polygonal basis as $\hat{S}_{\Delta}^{(n,k_1,k_2)}$ where $n = d_{\text{max}}$, $k_1 = d_{\text{max}}^x$ and $d_{\text{max}}^y$ or $n = \bar{d}_{\text{max}}$, $k_1 = \bar{d}_{\text{max}}^x$ and $k_2 = \bar{d}_{\text{max}}^y$ respectively.

We evaluate for required number of points $(x_i, y_j)$ the constant matrices $A(x, y)$ and we calculate (a) the corresponding determinants with interpolation set $(x_i, y_j, \det(A(x, y)))$ and (b) the corresponding inverse matrices with interpolation set $(x_i, y_j, A^{-1}(x, y))$.

We know that the inverse of a two variable polynomial matrix $A(x, y)$ is given by

$$A^{-1}(x, y) = \frac{\text{adj}A(x, y)}{\det A(x, y)}$$

where $\text{adj}A(x, y)$ and $\det A(x, y)$ is the adjoint matrix and the determinant of polynomial matrix $A(x, y)$ respectively. The adjoint matrix $\text{adj}A(x, y)$ is a two variable polynomial matrix and the determinant $\det A(x, y)$ is a two variable polynomial.

The adjoint matrix $\text{adj}A(x, y)$ is given by

$$\text{adj}A(x, y) = A^{-1}(x, y) \cdot \det A(x, y)$$

which for each interpolation point $(x_i, y_j)$ is reform

$$\text{adj}A(x_i, y_j) = A^{-1}(x_i, y_j) \cdot \det A(x_i, y_j) \quad (4)$$

**Step 4:** Compute the interpolating polynomial $p(x, y) = \det (A(x, y))$

$$p(x, y) = \det (A(x, y)) = X^T \cdot DD \cdot Y$$

where $X$, $DD$, $Y$ have been defined in [6].

**Step 5:** Compute the adjoint matrix of $A(x, y)$

$$\text{adj}A(x, y) = \bar{A}(x, y) = \bar{X}^T \cdot \bar{D} \cdot \bar{Y}$$

where $\bar{X}$, $\bar{D}$, $\bar{Y}$ have been defined in [6].

**Example 1** Let the polynomial matrix

$$A(x, y) = \begin{pmatrix}
-1 & 0 & x \\
5 & 1 & -1 \\
2 & 3xy & 2
\end{pmatrix}$$

**Step 1:** Computation of the degree of the determinant and the degree of each element of the adjoint matrix.

The above polynomial matrix has the degree matrices

$$D = \begin{pmatrix}
0 & 0 & 1 \\
0 & 0 & 0 \\
0 & 2 & 0
\end{pmatrix}, \ D^x = \begin{pmatrix}
0 & 0 & 1 \\
0 & 0 & 0 \\
0 & 1 & 0
\end{pmatrix}, \ D^y = \begin{pmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 1 & 0
\end{pmatrix}$$
We calculate the numbers \( d_{\text{min}} = \text{DegDet}(D) = 3, \) \( d_{\text{min}}^x = \text{DegDet}(D^x) = 2 \) and \( d_{\text{min}}^y = \text{DegDet}(D^y) = 1 \) which hold that \( d_{\text{min}} = d_{\text{min}}^x + d_{\text{min}}^y. \)

Sequentially, we calculate the matrices

\[
\bar{D} = \begin{pmatrix} 2 & 3 & 1 \\ 0 & 1 & 1 \\ 2 & 2 & 0 \end{pmatrix}, \quad \bar{D}^x = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}, \quad \bar{D}^y = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}
\]

and the corresponding numbers \( \bar{d}_{\text{max}} = 3, \bar{d}_{\text{max}}^x = 2, \bar{d}_{\text{max}}^y = 1 \) which hold that \( \bar{d}_{\text{max}} = \bar{d}_{\text{max}}^x + \bar{d}_{\text{max}}^y. \)

**Step 2:** Computation of the number of required points.

Then, the number of required interpolation points for the calculation of the determinant of \( A(x, y) \) is

\[
M_0 = (d_{\text{min}}^x + 1) (d_{\text{min}}^y + 1) = (2 + 1) (1 + 1) = 6
\]

and the corresponding set is given by \( \tilde{S}_{(2,1)}^{(2,1)} = \{(x_i, y_j)|i = 0, 1, 2, \quad j = 0, 1\}. \)

Accordingly, the number of required interpolation points for the calculation of the adjoint matrix of \( A(x, y) \) is

\[
N_0 = (\bar{d}_{\text{max}}^x + 1) (\bar{d}_{\text{max}}^y + 1) = (2 + 1) (1 + 1) = 6
\]

and the corresponding set is given by \( \tilde{S}_{(2,1)}^{(2,1)} = \{(x_i, y_j)|i = 0, 1, 2, \quad j = 0, 1\}. \)

**Step 3:** Evaluation part

We define the set of required points on orthogonal basis as

\[
\tilde{S}_{(2,1)}^{(2,1)} = \{(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1)\}
\]

We evaluate for required number of points \( (x_i, y_j) \) the constant matrices \( A(x_i, y_j) \) and we calculate (a) the corresponding determinants with interpolation set \( (x_i, y_j, \det(A(x_i, y_j))) \) and (b) the corresponding inverse matrices with interpolation set \( (x_i, y_j, A^{-1}(x_i, y_j)) \). For example

\[
A_{0,0} = A(x_0, y_0) = A(0, 0) = \begin{pmatrix} -1 & 0 & 0 \\ 5 & 1 & -1 \\ 2 & 0 & 2 \end{pmatrix}
\]

with determinant

\[
\det(A_{0,0}) = -2
\]

and inverse matrix

\[
A_{0,0}^{-1} = A^{-1}(0, 0) = \begin{pmatrix} -1 & 0 & 0 \\ 6 & 1 & \frac{1}{2} \\ 1 & 0 & \frac{1}{2} \end{pmatrix}
\]
With the same way we calculate for the rest point.

**Step 4:** Compute the interpolating polynomial \( p(x, y) = \det(A(x, y)) \)

\[
p(x, y) = \det(A(x, y)) = X^T \cdot DD \cdot Y = -2 - 2x - 3xy + 15x^2y
\]

**Step 5:** Compute the adjoint matrix of \( A(x, y) \)

\[
adjA(x, y) = \bar{A}(x, y) = \bar{X}^T \cdot \bar{D} \cdot \bar{Y} = \begin{pmatrix}
3xy + 2 & 3x^2y & -x \\
-12 & -2x - 2 & 5x - 1 \\
15xy - 2 & 3xy & -1
\end{pmatrix}
\]

The inverse matrix is given by

\[
A^{-1}(x, y) = \frac{1}{\det A(x, y)} \begin{pmatrix}
3xy + 2 & 3x^2y & -x \\
-12 & -2x - 2 & 5x - 1 \\
15xy - 2 & 3xy & -1
\end{pmatrix}
\]

3. On the parallel computation of the inverse of a two-variable polynomial matrix

The algorithm of the computation of the inverse of a two-variable polynomial matrix can be parallelized in two steps

- in evaluation part and
- in interpolation part

3.1 Parallelized evaluation part

In the evaluation part we need to calculate \( M_0 \) determinants and \( N_0 \) inverse matrices. We separate the calculations in the numbers of cores \( p \) of machine. Then, each core will calculate \( \frac{M_0}{p} \) determinants and \( \frac{N_0}{p} \) inverse matrices. For example if we have \( M_0 = 20 \), \( N_0 = 20 \) and \( p = 4 \) each core will calculate 5 determinants and 5 inverse matrices respectively.

In MATLAB we replace the loop `for` with `parfor` and we use arrays with constant dimensions and not sliding variables.

3.2 Parallelized interpolation part

In the interpolation part we need to calculate the \( k^{th} \) order of divided differences of a matrix for the determinant and the \( k^{th} \) order of divided differences of many matrices for the adjoint matrix. We separate the calculations in the numbers of cores \( p \) of machine.

In MATLAB we replace the loop `for` with `parfor` and we use arrays with constant dimensions and not sliding variables.
Table 1: Execution times (sec) of the computation of the inverse matrix with \( d = 1 \) (symbolic/parallel)

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.78387</td>
<td>5.77444</td>
<td>16.5622</td>
<td>93.385</td>
<td>1463.57</td>
</tr>
<tr>
<td>2</td>
<td>1.36445</td>
<td>2.47812</td>
<td>3.30513</td>
<td>7.43141</td>
<td>24.1033</td>
</tr>
</tbody>
</table>

3.3 Execution times

The parallel algorithm uses only numerical operations which can be implemented to any programming environment which supports numerical operations. Therefore, we develop the algorithm in the programming environment MATLAB and we compare the sequential algorithm and the parallel with the built-in symbolic function of MATLAB \textit{inv()}. This function uses symbolic operations to find the inverse matrix of a polynomial matrix.

In Table 1 we show the comparison in execution time between the sequential and parallel algorithm and the build-in symbolic function of the MATLAB \textit{inv()}. The performance tests have been implemented in a PC with the following specifications: Intel Core Quad CPU (Q9400) at 2600 GHz with 3.5 Gb RAM, and the release of MATLAB is R2010a.

The dataset of this comparison are polynomial matrices with dimensions \( m \times m \) and the upper bound of the degree for each element is \( d \). Each polynomial entry has random coefficients from the set \{0, 1, \ldots, 10\}.

The performance tests are implemented for \( m \geq 5 \) and \( d = 1 \). From cite it is known that for small matrices the symbolic function of MATLAB \textit{inv()} is better from numerical algorithm. The parallel numerical algorithm is better from symbolic function of MATLAB \textit{inv()} and the execution time for MATLAB function is increased very quickly, like exponential function, respect to \( m \) while the execution time for parallel numerical function is increased proportional respect to \( m \), like a linear function.

4 Conclusions

A parallel algorithm for the computation of the inverse matrix of a two-variable polynomial matrix are presented. The advantages of this algorithm are: a) the exclusive use of numerical operations which gives the opportunity to develop this parallel algorithm in any programming environment and b) the reduction of execution time instead to function of MATLAB \textit{inv()} which uses symbolic operations.

Acknowledgements. The authors wish to acknowledge financial support provided by the Research Committee of the Technological Education Institute of Central Macedonia, under grant SAT/IC/02122015-235/8.
References


https://doi.org/10.1023/b:mult.0000037345.60574.d4

Received: June 11, 2017; Published: July 5, 2017