Efficiently Evaluating a Polynomial Matrix

Technical Report

Brent M. Dingle

Texas A&M University

September 2004, November 2005

Abstract:

Interpolation techniques are often used in calculating the determinant of a (multivariate) polynomial matrix. These techniques usually require "putting in" a specified set of values for each variable found in the matrix, thus evaluating the matrix into a strictly numeric form. If the matrix is dense and the polynomials within the matrix have a large number of terms, evaluating the polynomial matrix at a specified set of points can be time consuming. This paper presents an efficient way of performing these evaluations under these circumstances based on a multidimensional matrix representation of the polynomials themselves. Effectively the matrix of polynomials becomes a matrix of matrices. This technique allows a large quantity of redundant calculations to be avoided and thus performs more quickly then a simple brute force method.

1 Introduction

There are a variety of circumstances when it may be necessary to evaluate a polynomial at a particular set of points in its variable space. This need often arises when attempting to interpolate a determinant of a large polynomial matrix. This is best illustrated with an example, note for simplicity the example is a small matrix, whereas in many scenarios it would be larger and the polynomials would be of higher degree and dense in terms.

Consider the polynomial matrix:

$$\begin{bmatrix} x^2+2xy-1 & 8y^2 & y+1 \\ x^2-3 & y & xy+y^2 \\ 7x^2+12xy-y & y^2+5 & 2xy \end{bmatrix}$$

The problem we wish to solve is to expedite the process of evaluating the matrix at a set of points. For example, evaluating the matrix at the points { (x, y) | x=1…20, y=4…7 }

2 Brute Force Algorithm

The most simple and easy to implement method of evaluating the matrix is to evaluate each polynomial entry individually and assign the result to the corresponding entry of a numeric matrix.

For instance, using the example given in section 1, the process would go something like:
Assume mat[80][3][3] is an array of eighty, 3 by 3 numeric matrices[*].
Assume sym_mat[3][3] is the polynomial matrix as given in section 1.

```
mat_index = 0;
For x = 1 to 20
   For y = 4 to 7
      For i = 0 to 2
         For j = 0 to 2
            mat[mat_index][i][j] = Eval Poly at sym_mat[i][j]
                                   with ‘x’ = x, ‘y’ = y
         End for j
      End for i
      mat_index = mat_index + 1
   End for y
End for x
```

Just for ONE of the eighty numeric matrices we would evaluate the value of x^2 three times, the value of y^2 three times and the value of xy four times. While this may seem insignificant, in larger cases with higher exponents and larger matrices this becomes a time consuming task.

[*]As a side note, usually all eighty numeric matrices are not stored, more often only the determinant or similar is kept. However, we will store the matrices themselves for the purpose of illustration.

3 Exponent Matrices

It is possible to avoid this redundancy of calculation with some precalculations and some clever data structures. Specifically let's turn each polynomial into a *k*-dimensional matrix, where *k* is the number of unique variables appearing in the polynomial matrix. So each variable represents a dimension in this new matrix. Thus, the size of each dimension of the matrix will be determined by the maximum exponent on the dimension's corresponding variable.

Again returning to the example of section 1. Suppose our polynomial matrix is:

$$\begin{bmatrix} x^2+2xy-1 & 8y^2 & y+1 \\ x^2-3 & y & xy+y^2 \\ 7x^2+12xy-y & y^2+5 & 2xy \end{bmatrix}$$

Then we will represent each polynomial entry as a 2-dimensional matrix, as there are 2 unique variables present, specifically *x* and *y*. Note the maximum degree of *x* is two and the maximum degree of *y* is two so the size in both dimensions is three. Thus the polynomial $x^2 + 2xy - 1$ becomes the coefficient matrix indexed by exponents:

$$x^2 + 2xy - 1 = \begin{array}{cc} & \begin{array}{ccc} y^0 & y^1 & y^2 \end{array} \\ \begin{array}{c} x^0 \\ x^1 \\ x^2 \end{array} & \begin{bmatrix} -1 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

And our polynomial matrix becomes the *CoeffEx Matrix* :

$$\begin{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} -3 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & -1 & 0 \\ 0 & 12 & 0 \\ 7 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 5 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix}$$

From this we know what combinations of powers of *x* and *y* will be needed. Specifically we can consider 0 as false and anything else as true. We can then "OR" the above 9 submatrices together to obtain a *Boolean power matrix*:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Thus we need calculate only the combination of powers of *x* and *y* that are used, and we only need to calculate them once. In the case above we must find: x^0y^0, y^1, x^1y^1, x^2, y^2.

Notice that there is an obvious dependency in the *Boolean power matrix*. For i > 0 and j > 0 the element [i][j] = [i][j-1] * [i-1][j]. Thus in the case above, because we need to find x^1y^1 we must find x^1 and y^1. Thus the final *Boolean power matrix* that would be used would be of the form:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Further analysis of this matrix structure and its dependencies may offer speed increases beyond what we will discuss here. For simplicity we will be using it only to determine which powers need to be calculated.

Going back to our example and using the above *Boolean power matrix* we see we must calculate x^0, y^0, x^1, y^1, x^2, and y^2. From this we will create the matrix:

$$\begin{bmatrix} 1 & y & y^2 \\ 0 & xy & 0 \\ x^2 & 0 & 0 \end{bmatrix}$$

This matrix will be numeric as when it is created we will know the value of *x* and *y*. More importantly as will be shown below it will be created only once for each (*x*, *y*) pair.

4 Submatrix Algorithm

So using the above *CoeffEx and Boolean power matrices* we can alter our basic algorithm as presented in section 1 to the following:

```
Input: A polynomial matrix named sym_mat and a range of values
    for x and y = {(x,y) | x_min ≤ x ≤ x_max, y_min ≤ y ≤ y_max }
Output: An array of numeric matrices each representing the result
    of substituting in an (x,y) pair into sym_mat.
```

```
Create the CoeffEx matrice for sym_mat, name it coeffex_mat.
              Note each entry in coeffex_mat is a submatrix.
Create the corresponding Boolean power matrix, name it bpow_mat.
mat_index = 0;
For x = x_min to x_max
   Based on bpow_mat calculate the needed powers of x.
   Store the results in pow_mat.
   For y = y_min to y_max
      Based on bpow_mat calculate the needed powers of y.
      Store the results in pow_mat.
      Complete the required entries of pow_mat.
      For i = 0 to 2
         For j = 0 to 2
            mat[mat_index][i][j]= Dot(pow_mat, coeffex_mat[i][j])
         End for j
      End for i
      mat_index = mat_index + 1
   End for y
End for x
```

Notice in the above the function dot performs an element-wise multiply of each matrix and returns the sum of the numbers. For example:

$$Dot\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \begin{bmatrix} 10 & 70 & 40 \\ 20 & 80 & 50 \\ 30 & 90 & 60 \end{bmatrix}\right) = \text{Sum of entries of} \begin{bmatrix} 10 & 140 & 120 \\ 80 & 400 & 300 \\ 210 & 720 & 540 \end{bmatrix}$$

$$=10+140+120+80+400+300+210+720+540$$

$$= 2520$$

5 Complete Example

For illustrative purposes we will now demonstrate a complete example showing all the steps, for one (x, y) pair, specifically (5, 7).

$$\text{Let } sym\_mat = \begin{bmatrix} x^2+2xy-1 & 8y^2 & y+1 \\ x^2-3 & y & xy+y^2 \\ 7x^2+12xy-y & y^2+5 & 2xy \end{bmatrix}$$

Then *coeffex_mat* = $\begin{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} -3 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & -1 & 0 \\ 0 & 12 & 0 \\ 7 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 5 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix}$

And *bpow_mat* = $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$

Assume we are at the point $x = 5$ and $y = 7$ and *mat_index* = 12.
Using *bpow_mat* we would calculate *pow_mat* to be:

pow_mat = $\begin{bmatrix} 1 & 7 & 49 \\ 5 & 35 & 0 \\ 25 & 0 & 0 \end{bmatrix}$

We now enter the for-i and for-j loops of the algorithm.

$$mat[12][0][0] = Dot\left(\begin{bmatrix} 1 & 7 & 49 \\ 5 & 35 & 0 \\ 25 & 0 & 0 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}\right)$$

$$= \text{Sum of entries of } \begin{bmatrix} -1 & 0 & 0 \\ 0 & 70 & 0 \\ 25 & 0 & 0 \end{bmatrix}$$

$= -1 + 70 + 25$

$= 94$

Note if $f(x,y) = x^2 + 2xy - 1$, then $f(5, 7) = 25 + 70 - 1 = 94$.

$$mat[12][0][1] = Dot\left(\begin{bmatrix} 1 & 7 & 49 \\ 5 & 35 & 0 \\ 25 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right)$$

$$= \text{Sum of entries of} \begin{bmatrix} 0 & 0 & 392 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$= 392$$

$$mat[12][0][2] = Dot\left(\begin{bmatrix} 1 & 7 & 49 \\ 5 & 35 & 0 \\ 25 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right)$$

$$= \text{Sum of entries of} \begin{bmatrix} 1 & 7 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$= 8$$

$$mat[12][1][0] = Dot\left(\begin{bmatrix} 1 & 7 & 49 \\ 5 & 35 & 0 \\ 25 & 0 & 0 \end{bmatrix}, \begin{bmatrix} -3 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}\right)$$

$$= \text{Sum of entries of} \begin{bmatrix} -3 & 0 & 0 \\ 0 & 0 & 0 \\ 25 & 0 & 0 \end{bmatrix}$$

$$= 22$$

$$mat[12][1][1] = Dot\left(\begin{bmatrix} 1 & 7 & 49 \\ 5 & 35 & 0 \\ 25 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right)$$

$$= \text{Sum of entries of} \begin{bmatrix} 0 & 7 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$= 7$$

$$mat[12][1][2] = Dot\left(\begin{bmatrix}1 & 7 & 49\\ 5 & 35 & 0\\ 25 & 0 & 0\end{bmatrix}, \begin{bmatrix}0 & 0 & 1\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}\right)$$

$$= \text{Sum of entries of} \begin{bmatrix}0 & 0 & 49\\ 0 & 35 & 0\\ 0 & 0 & 0\end{bmatrix}$$

$$= 84$$

$$mat[12][2][0] = Dot\left(\begin{bmatrix}1 & 7 & 49\\ 5 & 35 & 0\\ 25 & 0 & 0\end{bmatrix}, \begin{bmatrix}0 & -1 & 0\\ 0 & 12 & 0\\ 7 & 0 & 0\end{bmatrix}\right)$$

$$= \text{Sum of entries of} \begin{bmatrix}0 & -7 & 0\\ 0 & 420 & 0\\ 175 & 0 & 0\end{bmatrix}$$

$$= 588$$

$$mat[12][2][1] = Dot\left(\begin{bmatrix}1 & 7 & 49\\ 5 & 35 & 0\\ 25 & 0 & 0\end{bmatrix}, \begin{bmatrix}5 & 0 & 1\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\right)$$

$$= \text{Sum of entries of} \begin{bmatrix}5 & 0 & 49\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$$

$$= 54$$

$$mat[12][0][2] = Dot\left(\begin{bmatrix}1 & 7 & 49\\ 5 & 35 & 0\\ 25 & 0 & 0\end{bmatrix}, \begin{bmatrix}0 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 0\end{bmatrix}\right)$$

$$= \text{Sum of entries of} \begin{bmatrix}0 & 0 & 0\\ 0 & 70 & 0\\ 0 & 0 & 0\end{bmatrix}$$

$$= 70$$

So in the end $mat[12] = \begin{bmatrix} 94 & 392 & 8 \\ 22 & 7 & 84 \\ 588 & 54 & 70 \end{bmatrix}$

Notice the determinant of *mat*[12] is 18,354,236.

As a check notice the determinant of the original symbolic matrix is:

$$-15 + 5x^2 - 15y + 5xy - 2x^2y - 5x^3y + 3y^2 - 14xy^2 - 21x^2y^2 + 2x^3y^2 - 2y^3 + 27xy^3 + 5x^2y^3 + 39x^3y^3 + y^4 - 8xy^4 + 149x^2y^4 - 8y^5 + 94xy^5$$

Which evaluated at $x = 5$ and $y = 7$ is 18,354,236 and thus everything comes together nicely.

6 Conclusion

We have now shown an algorithm demonstrating that the simple brute force algorithm to evaluate a matrix of polynomials is not the most efficient method. We have also presented a much more efficient algorithm. From this it should be clear that a significant speed increase can be achieved simply using clever data structures and well planned arithmetic.

This method of evaluation would be extremely useful in cases where an interpolating method of finding the determinant of a symbolic matrix was necessary. This method may also prove useful in other related scenarios.

None of this is of revolutionary nature, but is simply stated here as a reference and guide for future work in numerical and symbolic programming endeavors.