An efficient algorithm for evaluating polynomials in the Pólya basis

Joe Warren
Department of Computer Science
Rice University

October 1995

Abstract

A new $O(n)$ algorithm is given for evaluating univariate polynomials of degree n in the Pòlya basis. Since the Lagrange, Bernstein, and monomial bases are all special instances of the Pòlya basis, this technique leads to efficient evaluation algorithms for these special bases. For the monomial basis, this algorithm is shown to be equivalent to Horner's rule.

1 Pòlya basis functions

Let $n_k(t)$ be the restriction of the standard B-spline basis functions [dB72] of degree n over the nondecreasing knot sequence $t_1, t_2, ..., t_{2n}$ to the interval $t_n < t < t_{n+1}$. The Pòlya basis functions are the unique polynomial functions $d_k(t)$ that satisfy Marsden's identity [Mar70].

$$(x-t)^n = \sum_{k=0}^{n} d_k(t) n_k(x).$$

The Pòlya basis functions of degree n can be written explicitly as

$$d_k(t) = \prod_{i=1}^{n} (t_{k+i} - t). \tag{1}$$

(See Barry and Goldman [BG93, pp.28] for more details.)

Using the explicit definition of equation 1, the Pòlya functions can be defined directly for arbitrary sequences $t_1, ..., t_{2n}$. Under this definition, the Pòlya functions includes several important bases as special cases. These bases are:

- The Lagrange basis,

- The Bernstein basis,
- The monomial basis.

Given a set of $n+1$ distinct values $u_0, ..., u_n$, The Lagrange basis functions $l_k(t)$ [BF85] of degree n are the $n+1$ polynomial basis functions that satisfy

$$l_k(u_i) = \delta_{ki}.$$

The kth Lagrange basis function can be explicitly written as

$$l_k(t) = \frac{1}{\alpha_k} \prod_{i \neq k} (u_i - t)$$

where $\alpha_k = \prod_{i \neq k}(u_i - u_k)$. The Lagrange basis may be viewed as an instance of the Pòlya basis. Consider the Pòlya basis with knot sequence

$$\begin{array}{llll} t_1 = u_1, & t_2 = u_2, & ..., & t_n = u_n, \\ t_{n+1} = u_0, & t_{n+2} = u_1, & ..., & t_{2n} = u_{n-1}. \end{array}$$

The Pòlya basis function $d_k(t)$ is exactly the Lagrange basis function $l_k(t)$ multiplied by the constant α_k.

The Bernstein basis functions of degree n are

$$b_k(t) = \frac{n!}{k!(n-k)!} t^k (1-t)^{n-k}.$$

These basis functions are the building blocks for one of the most popular curve representations in geometric design, Bézier curves [Far88]. Specializing the knot sequence to

$$\begin{array}{llll} t_1 = 0, & t_2 = 0, & ..., & t_n = 0, \\ t_{n+1} = 1, & t_{n+2} = 1, & ..., & t_{2n} = 1 \end{array}$$

yields a Pólya basis of the form

$$d_k(t) = (-t)^{n-k}(1-t)^k.$$

Thus, the Pòlya basis function $d_k(t)$ is exactly the Bernstein basis function $b_{n-k}(t)$ multiplied by the constant $(-1)^{n-k}\frac{n!}{k!(n-k)!}$.

Finally, the monomial basis, $\{1, t, t^2, ..., t^n\}$ is probably the most commonly used basis in mathematics. To express the monomial basis as a Pòlya basis, one must first develop a

notion of a knot at infinity. The affine knot t_i can be homogenized to yield the homogeneous representation $(t_i, 1)$. The linear factors in the Pòlya basis can be expressed in the form $(t_i * 1 - 1 * t)$. If $t_i = \infty$, then the corresponding homogeneous knot is $(1, 0)$. The related linear factor is $(1 * 1 - 0 * t)$, the constant 1. (For information on this interpretation, see [BG93].) Thus, specializing the knot sequence to have values

$$\begin{array}{llll} t_1 = 0, & t_2 = 0, & ..., & t_n = 0, \\ t_{n+1} = \infty, & t_{n+2} = \infty, & ..., & t_{2n} = \infty \end{array}$$

yields Pólya basis functions of the form

$$d_k(t) = t^{n-k}.$$

These basis function are exactly the monomial basis function after renumbering.

2 Evaluation in the Pòlya basis

Let $d(t)$ be a polynomial of degree n in the Pòlya basis.

$$d(t) = \sum_{k=0}^{n} P_k d_k(t). \tag{2}$$

We next describe an algorithm for evaluating $d(t)$ that requires $O(n)$ arithmetic operations per evaluation. The following recurrence describes such an algorithm.

$$\begin{aligned} L_0 &= 1, \\ U_0 &= P_0, \\ L_{i+1} &= L_i * (t_{i+n+1} - t), \\ U_{i+1} &= U_i * (t_{i+1} - t) + P_{i+1} * L_{i+1}. \end{aligned} \tag{3}$$

This recurrence can be expressed as a ladder shaped diagram in figure 1. Values are associated with the intersection of arrows and flow up the ladder. A value at an intersection is passed along an outgoing edge and multiplied by the label of that edge. The resulting value is then added to the value at the end of edge. The $L_i's$ lie along the lower right portion of the ladder. The $U_i's$ lie along the upper left portion of the ladder.

Theorem 1 *If $d(t)$ is defined as in equation 2 and U_n is defined in equation 3, then*

$$U_n = d(t).$$

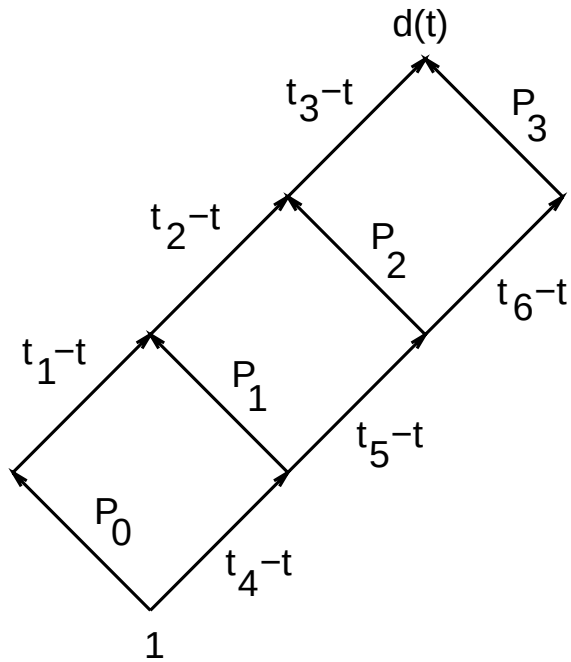

Figure 1 The ladder recurrence for $n = 3$

Proof: Obviously, L_j satisfies

$$L_j = \prod_{i=1}^{j}(t_{i+n} - t). \tag{4}$$

We claim that U_j satisfies

$$U_j = \sum_{i=0}^{j} P_i(\prod_{\alpha=i+1}^{j}(t_\alpha - t))(\prod_{\beta=1}^{i}(t_{\beta+n} - t)).$$

The proof is by induction on j. For the base case $j = 0$, this expression evaluates to P_0 exactly as specified by the recurrence. If this expression is true for U_j, then we next show that also is true for U_{j+1}. By equation 3,

$$U_{j+1} = U_j * (t_{j+1} - t) + P_{j+1} * L_{j+1}.$$

Using the inductive hypothesis and replacing L_{j+1} by the right hand side of equation 4 yields

$$U_{j+1} = (\sum_{i=0}^{j} P_i(\prod_{\alpha=i+1}^{j}(t_\alpha - t))(\prod_{\beta=1}^{i}(t_{\beta+n} - t))) * (t_{j+1} - t) + P_{j+1} * (\prod_{\beta=1}^{j+1}(t_{\beta+n} - t).$$

After simplification, this relation establishes the inductive hypothesis for U_{j+1}.

To conclude the proof, we note that if $j = n$, then U_n agrees exactly with the definition of $d(t)$. □

3 Comparison with other methods

Given a set of $n+1$ coefficients for a polynomial in the Pòlya basis, evaluating this recurrence requires $O(n)$ additions and multiplications. Note that this method avoids any divisions by functions of t. To evaluate a polynomial in the Lagrange basis using this recurrence, one must first divide the kth coefficient in the Lagrange basis by α_k and then apply the recurrence. All of the α_k's can be precomputed once in an $O(n)$ preprocessing step and subsequently looked up. The speed of this method compares favorably with that of Neville's algorithm [BF85], the standard evaluation recurrence for the Lagrange basis that requires $O(n^2)$ operations per evaluation.

The standard evaluation method for polynomials in the Bernstein basis is the deCasteljau algorithm [Far88]. This algorithm requires $O(n^2)$ operations per evaluation. To evaluate a polynomial in the Bernstein basis using the ladder recurrence, the coefficient of $b_k(t)$ is first multiplied by $(-1)^{n-k}\frac{n!}{k!(n-k)!}$ and then used as input to the ladder recurrence. Since these binomial coefficients can be precomputed in $O(n)$ time, evaluation using the ladder recurrence requires only $O(n)$ time per evaluation.

In the case of the monomial basis, the knots $t_{n+1}, ..., t_{2n}$ were placed at infinity. The linear factors corresponding to these knots were the constant functions 1. Therefore, all the terms L_i in the ladder recurrence were also one and the lower left side of the recurrence in figure 1 could be deleted. The resulting recurrence on the U_j's satisfies

$$\begin{aligned} U_0 &= P_0, \\ U_{i+1} &= U_i * t + P_{i+1}. \end{aligned}$$

This recurrence is exactly Horner's rule for evaluating the polynomial

$$\sum_{j=0}^{n} P_{n-k} t^k$$

in the monomial basis [BF85].

4 Conclusion

The existence of $O(n)$ evaluation methods for polynomials in the Lagrange and Bernstein basis is not surprising in itself. For the Lagrange basis, building such a method is simple if one

is allowed to resort to division by $(u_k - t)$. In the case of the Bernstein basis, a table of various powers of t and $(1-t)$ can be used to evaluate the basis functions efficiently. The beauty of this result lies in the fact that the ladder recurrence provides a natural generalization of Horner's method for the monomial basis to polynomials in the Lagrange and Bernstein basis. The recurrence provides an efficient method for evaluating a polynomial in the Pòlya basis without resorting to division by polynomial functions in t (and the numerical difficulties associated with such an approach). For the Lagrange basis, the recurrence is asymptotically faster than Neville's algorithm. For the Bernstein basis, the recurrence is asymptotically faster than the DeCasteljau algorithm. One remaining question is the stability of the ladder recurrence, especially in relation to the methods described above. The author hopes to address this problem in future work.

References

[BF85] R. Burden and J. D. Faires. *Numerical Analysis.* Prindle,Weber, and Schmidt, 1985.

[BG93] P. Barry and R. Goldman. Algorithms for progressive curves. In R. Goldman and T. Lyche, editors, *Knot Insertion and Deletion Algorithms for B-Spline Curves and Surfaces*, pages 11–63. SIAM, 1993.

[dB72] Carl de Boor. *A Practical Guide to Splines.* Springer-Verlag, 1972.

[Far88] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Academic Press Inc., New York, 1988.

[Mar70] M. J. Marsden. An identity function for spline functions with applications to variation-diminishing spline approxiamtion. *Journal of Approximation Theory*, 3:7–49, 1970.