Polynomials, Newton Form and Nested Multiplication

By

Dr. W. Dan Curtis

Introduction

The falling factorials \( x^m \) for \( m \) a non-negative integer are defined by:

\[
x^0 = 1 \\
x^m = x(x-1)(x-2)\cdots(x-(m-1)) \quad \text{for } m \geq 1.
\]

Therefore \( x^m \) is a polynomial of degree \( m \).

We easily see for small \( m \) that any polynomial of degree \( \leq m \) can be expressed as a linear combination of \( x^k \) for \( 0 \leq k \leq m \).

**Exercise 1**: Express \( p(x) = x^2 - x + 1 \) in terms of \( x^0, x^1, \) and \( x^2 \).

Is there a systematic way of doing this? That is, could we write a computer program to do it? We shall answer this question and many others in what follows.

Representation of Polynomials – the Newton Form

Normally, we write polynomials in terms of powers of the variable (which we will take to be \( x \)):

\[
p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1} + a_nx^n
\]

This form is inefficient if we want to evaluate the polynomial using only addition and multiplication.

**Exercise 2**: Let \( c_n \) be the total number of additions and multiplications required to evaluate a polynomial of degree \( n \) (assuming all coefficients are non-zero). Then \( c_0 = 0, c_1 = 2, c_2 = 5, c_3 = 9 \) (verify!). Show that the \( c_n \) satisfy the recursion

\[
c_n = c_{n-1} + n + 1
\]

and use this to compute \( c_{10} \).
We can dramatically improve the efficiency of polynomial evaluation by rewriting in the so-called **nested form**:

\[ p(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_n))) \cdots \]

Evaluation of \( p(c) \) is now accomplished by the following loop:

\[
\begin{align*}
val &= a_n; \\
k &= n; \\
while(k > 0) \\
& \{
val = c*val + a_{k-1}; \\
k &= -;
\}
\end{align*}
\]

**Exercise 3:** Let \( d_n \) be the total number of additions and multiplications used in evaluating a polynomial of degree \( n \) using the above algorithm. Find the value \( d_{10} \) and compare with the result of Exercise 2.

A more general way of writing a polynomial is the **Newton form**: The polynomial in Newton form having **coefficients** \( a_0, a_1, \cdots, a_n \) and **centers** \( c_1, c_2, \cdots, c_n \), is the polynomial

\[ p(x) = a_0 + a_1(x - c_1) + a_2(x - c_1)(x - c_2) + \cdots + a_n(x - c_1)(x - c_2)\cdots(x - c_n) \]

**Example 1:**

a. The polynomial \( p(x) = 2 - 3x + 4x^2 + 2x^4 \) is in Newton form with coefficients \( 2, -3, 4, 0, 2 \) and centers \( 0, 0, 0, 0 \).

b. The polynomial \( q(x) = 2 - 5(x + 1) + (x + 1)^3 + 2(x + 1)^4 \) is in Newton form with coefficients \( 2, -5, 0, 1, 2 \) and centers \( -1, -1, -1, -1 \).
c. The polynomial \( p(x) = 1 - 5x + 3x(x - 1) + 2x(x - 1)(x - 2) \) is in Newton form with coefficients

\[
1, -5, 3, 2
\]

and centers

\[
0, 1, 2
\]

The following algorithm is the main reason we have introduced the Newton form.

**Theorem** (Nested Multiplication Algorithm) Let \( p(x) \) be the polynomial in Newton form having coefficients \( a_0, a_1, \ldots, a_n \) and centers \( c_1, c_2, \ldots, c_n \) and let \( z \in \mathbb{R} \). Define \( a_0', a_1', \ldots, a_n' \) by

\[
a_n' = a_n
\]

\[
\text{for}(i = n - 1; i \geq 0; i - -)
\]

\[
\{ a_i' = a_i + (z - c_{i+1})a_{i+1}'
\}

Then \( a_0' = p(z) \) and the polynomial in Newton form with coefficients \( a_0', a_1', \ldots, a_n' \) and centers \( z, c_1, \ldots, c_{n-1} \) is the polynomial \( p(x) \).

**Proof**: We proceed by induction on \( n \). The case \( n = 0 \) is trivial. Let us verify the case \( n = 1 \). The algorithm reads

\[
a_1' = a_1
\]

\[
a_0' = a_0 + (z - c_1)a_1'
\]

Therefore we have

\[
p(x) = a_0 + a_1(x - c_1)
\]

\[
= a_0' - a_1'(z - c_1) + a_1'(x - c_1)
\]

\[
= a_0' + a_1'(x - z)
\]

as asserted.

For the inductive step, we assume we know that the theorem is true for \( n - 1 \) and we prove it for \( n \). So assume that \( p(x) \) can be written in Newton form with coefficients \( a_0, a_1, \ldots, a_n \) and centers \( c_1, c_2, \ldots, c_n \). Then we can write

\[
p(x) = a_0 + (x - c_1)q(x)
\]
where \( q(x) \) is in Newton form with coefficients \( a_1, \ldots, a_n \) and centers \( c_2, \ldots, c_n \). Now we apply the algorithm and create the numbers \( a'_1, \ldots, a'_n \). By the induction hypothesis, \( q(x) \) can be written in Newton form with coefficients \( a'_1, \ldots, a'_n \) and centers \( z, c_2, \ldots, c_{n-1} \). Therefore we have

\[
p(x) = a_0 + (x - c_1)(a'_1 + a'_2(x - z) + a'_3(x - z)(x - c_2) + \cdots)
\]

\[
= a'_0 - (z - c_1)a'_1 + (x - c_1)a'_1 + a'_2(x - z)(x - c_2) + \cdots
\]

\[
+ a'_n(x - z)(x - c_1)(x - c_2) \cdots (x - c_{n-1})
\]

\[
= a'_0 + (x - z)a'_1 + a'_2(x - z)(x - c_1) + \cdots
\]

\[
+ a'_n(x - z)(x - c_1)(x - c_2) \cdots (x - c_{n-1})
\]

as asserted. □

**Applications of the Nested Multiplication Algorithm (NMA)**

The NMA provides an efficient algorithm for evaluating a polynomial written in Newton form.

**Exercise 4:**

a. Let \( w_n \) be the total number of additions and multiplications used in evaluating a polynomial of degree \( n \) written in Newton form. Find a recursion formula relating \( w_n \) and \( w_{n-1} \).

b. Find the number of floating point operations used in evaluating a polynomial of degree 10.

**Example 2:** The falling factorial \( x^m \) is the polynomial

\[
x^m = x(x - 1)(x - 2) \cdots (x - m + 1).
\]

Therefore, the problem of expressing a polynomial \( p(x) \) of degree \( \leq m \) in terms of the falling factorials \( x^k, 0 \leq k \leq m \) is equivalent to writing \( p(x) \) in Newton form with centers \( 0, 1, \ldots, m - 1 \). Therefore, we start with the coefficients of \( p(x) \) and the list of centers \( 0, 0, \ldots, 0 \), and apply the NMA first with \( z = m - 1 \), then with \( z = m - 2 \) and continuing until we finally use \( z = 0 \).
We work out the specific case of writing \( p(x) = 1 - 2x + x^2 \) in terms of \( x^0, x^1, \) and \( x^2 \). That is we want to write \( p(x) \) as

\[
p(x) = b_0 1 + b_1 x + b_2 x(x - 1).\]

Beginning with the coefficients 1, -2, 1 and centers 0, 0, we apply NMA with \( z = 1 \):

\[
a_2' = 1 \\
 a_1' = -2 + (1 - 0)1 = -1 \\
 a_0' = 1 + (1 - 0)(-1) = 0
\]

Thus we have \( p(x) = (-1)(x - 1) + (1)(x - 1)(x) \). Then we do NMA again, this time with \( z = 0 \):

\[
a_2' = 1 \\
 a_1' = -1 + (0 - 0)1 = -1 \\
 a_0' = 0 + (0 - 1)(-1) = 1
\]

Thus, we have \( p(x) = 1 + (-1)x + (1)x(x - 1) \) as desired.

**Exercise 5:**

a. Express \( x^3 \) in terms of falling factorials.
b. Use the result of a. to find a function \( g(x) \) such that \( \Delta g(x) = x^3 \)
c. Find a formula for \( \sum_{k=1}^{n} k^3 \).
d. Write the polynomial \( p(x) = 2 - 4x + 3x^2 + x^4 \) in terms of powers of \( x - 2 \), that is, write

\[
p(x) = b_0 + b_1 (x - 2) + b_2 (x - 2)^2 + b_3 (x - 2)^3 + b_4 (x - 2)^4.
\]