General Polynomial Interpolation in Quadratic Time

Problem Let $A(x)$ be a polynomial of degree-bound n, and let $(\boldsymbol{x}, \boldsymbol{y})$ an n-point representation of $A(x)$. Use Lagrange's formula:

$$A(x) = \sum_{k=0}^{n-1} y_k \left[\frac{\prod_{j=0, j\neq k}^{n-1} (x - x_j)}{\prod_{j=0, j\neq k}^{n-1} (x_k - x_j)} \right] \tag{1}$$

to devise an algorithm to compute the vector $\boldsymbol{a}$ of coefficients of $A(x)$ in $O(n^2)$ time.

Solution: In order to compute $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1})$, we perform the following steps, which derive $\boldsymbol{a}$ from Formula (1).

1. Compute the coefficient vector $\boldsymbol{q} = (q_0, q_1, \ldots, q_n)$ of the $(n+1)$-degree bound polynomial
$$Q(x) = \prod_{j=0}^{n-1} (x - x_j).$$
2. For $0 \leq k \leq n-1$, obtain the coefficient vector $\boldsymbol{q}^{(k)} = (q_0^{(k)}, q_1^{(k)}, \ldots, q_{n-1}^{(k)})$ of the n-degree bound polynomial
$$Q^{(k)}(x) = \prod_{j=0, j\neq k}^{n-1} (x - x_j) = Q(x)/(x - x_k).$$
3. For $0 \leq k \leq n-1$, evaluate $Q^{(k)}(x)$ on x_k. Let $z_k = Q^{(k)}(x_k)$.
4. By observing that Formula (1) implies that
$$A(x) = \sum_{k=0}^{n-1} \frac{y_k}{z_k} Q^{(k)}(x),$$
for $0 \leq i \leq n-1$, obtain a_i as
$$a_i = \sum_{k=0}^{n-1} \frac{y_k q_i^{(k)}}{z_k}.$$

Let us now give the details for implementing the four steps described above.

Step 1 We can design a simple divide-and-conquer algorithm for computing $Q(x)$ based on the following property. For the base case $n = 1$ we observe that $\boldsymbol{q} = (q_0, q_1) = (-x_0, 1)$. For $n > 1$, assume that we have (recursively) computed the coefficients $\boldsymbol{q'}$ of

$$Q'(x) = \prod_{j=0}^{n-2} (x - x_j),$$

that is, we have solved the sub-instance of size $n-1$ consisting of multiplying the first $n-1$ degree-one polynomials. Then, for $0 \le i \le n$ we can obtain the coefficient q_i of $Q(x) = Q'(x)(x - x_{n-1})$ as

$$q_i = \begin{cases} -q'_0 x_{n-1} & i = 0, \\ q'_{n-1} & i = n, \\ q'_{i-1} - q'_i x_{n-1} & 0 < i < n. \end{cases} \tag{2}$$

Note that the above formula is a simple mathematical transcription of the elementary algorithm for multiplying two polynomials by cross-multiplying their component monomials and summing together the coefficients of the resulting monomials of equal degree. The algorithm is

```
COMPUTE_Q(x)
n ← length(x)
if n = 1 then return (−x_0, 1)
⋆ x' = (x_0, x_1, ... x_{n−2}) ⋆
q' ← COMPUTE_Q(x')
q_0 ← −q'_0 x_{n−1}
q_n ← q'_{n−1}
for i ← 1 to n − 1 do
    q_i ← q'_{i−1} − q'_i x_{n−1}
return q
```

The time complexity $T(n)$ of the above algorithm, whose correctness follows from Formula (2), obeys the recurrence

$$T(n) = \begin{cases} 1 & n = 1, \\ T(n-1) + cn & n > 1, \end{cases}$$

for a suitable constant $c > 1$. It is easy to check that $T(n) = \Theta(n^2)$.

Step 2 Let us fix k, with $0 \le k \le n-1$, and consider $Q^{(k)}(x) = Q(x)/(x - x_k)$. Since

$$Q(x) = Q^{(k)}(x)(x - x_k) \tag{3}$$

we can obtain a simple linear system of equations on the (unknown) coefficients $(q_0^{(k)}, q_1^{(k)}, \ldots, q_{n-1}^{(k)})$ of $Q^{(k)}(x)$ by imposing equality of the coefficients of the monomials of same degree of the two polynomial expressions at the left and right side of Equation (3). We have:

$$\begin{aligned} q_n &= q_{n-1}^{(k)} \\ q_{n-1} &= q_{n-2}^{(k)} - x_k q_{n-1}^{(k)} \\ &\vdots \\ q_{n-i} &= q_{n-i-1}^{(k)} - x_k q_{n-i}^{(k)}, \text{ for } 2 \le i \le n-1 \\ &\vdots \end{aligned}$$

We can solve the above simple system of n equations to compute the unknowns $q_i^{(k)}$, $0 \le i \le n-1$. This can be easily done by first setting $q_{n-1}^{(k)} = q_n$ and then obtaining $q_{n-2}^{(k)}$ as a function of q_{n-1} and $q_{n-1}^{(k)}$, $q_{n-3}^{(k)}$ as a function of q_{n-2} and $q_{n-2}^{(k)}$, and so on for all the other indices. (Note that this approach is similar to the one at the base of the standard grade-school polynomial division algorithm).

The algorithm implementing the above argument is the following:

DIVIDE($\boldsymbol{q}, x_k$)
$n \leftarrow \text{length}(\boldsymbol{q}) - 1$
$q'_{n-1} \leftarrow q_n$
for $i \leftarrow 2$ **to** n **do**
    $q'_{n-i} \leftarrow q_{n-i+1} + x_k q'_{n-i+1}$
return $\boldsymbol{q'}$

The loop body is executed $(n-1)$ times, so the running time is $\Theta(n)$. Observe that we need to invoke algorithm DIVIDE n times (one for each input $(\boldsymbol{q}, x_k), 0 \le k \le n-1$), hence Step 2 requires a total of $O(n^2)$ time.

Step 3 and Step 4 Recall that algorithm HORNER($\boldsymbol{a}, x$) (seen in class), evaluates a polynomial $A(x)$ of degree bound n and coefficient representation $\boldsymbol{a}$ on x in $O(n)$ time. By calling HORNER n times again on inputs $(\boldsymbol{q}^{(k)}, x_k)$, for $0 \le k \le n-1$, we obtain the necessary values $z_k = Q^{(k)}(x_k)$ in time $O(n^2)$. At this point, for $0 \le i \le n-1$, we obtain the values a_i by accumulating the entries $y_k q_i^{(k)}/z_k$ as specified in Step 4. Overall, the accumulations also require $O(n^2)$ time.

We are ready to give the code for the entire interpolation process. We will make use of the subroutines developed above for the single steps.

INTERPOLATE($\boldsymbol{x}, \boldsymbol{y}$)
$n \leftarrow \text{length}(\boldsymbol{x})$
$\boldsymbol{q} \leftarrow$ COMPUTE_Q($\boldsymbol{x}$)
for $i \leftarrow 0$ **to** $n-1$ **do** $a_i \leftarrow 0$
for $k \leftarrow 0$ **to** $n-1$ **do**
    $\boldsymbol{q'} \leftarrow$ DIVIDE($\boldsymbol{q}, x_k$)
    $z_k \leftarrow$ HORNER($\boldsymbol{q'}, x_k$)
    for $i \leftarrow 0$ **to** $n-1$ **do**
        $a_i \leftarrow a_i + y_k q'_i / z_k$
return $\boldsymbol{a}$

By combining the analyses for the single steps discussed above we can argue that Algorithm INTERPOLATE correctly computes the coefficient vector $\boldsymbol{a}$ in overall time $O(n^2)$ and $O(n)$ space. □