CS323 LECTURE NOTES - LECTURE 5

1 Horner's Method for Polynomial Evaluation

In the case of n-degree polynomials

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \quad (1)$$

our objective will be to find real roots, i.e. $r \in \mathbb{R}$ such that $P(r) = 0$.

Newton's Method can be easily used in this case since polynomial derivatives are easy to compute:

$$P'(x) = a_1 + 2a_2 x + 3a_3 x^2 + 4a_4 x^3 + \ldots + na_n x^{n-1}$$

Therefore, given a first approximation to the root we have that

$$r_1 = r_0 - \frac{P(r_0)}{P'(r_0)}$$

If we use (1) to compute $P(r_0)$ exactly as written we need n products to compute the last term, $n-1$ product to compute the second to last term and so on. Therefore, the total number of products required to compute it is

$$\sum_{i=1}^{n-1} = \frac{n(n-1)}{2}$$

It will also require $n-1$ additions, so we have that the total number of operations required to compute $P(r_0)$ is

$$T(n) = \frac{1}{2}n(n-1) + n \in \Theta(n^2)$$

It is possible to find a more efficient way to compute $P(r_0)$ using factorization. To illustrate this, let us use the following 4-th degree polynomial

$$P(r_0) = a_0 + a_1 r_0 + a_2 r_0^2 + a_3 r_0^3 + a_4 r_0^4$$

we can factor it in the following way

$$P(r_0) = a_0 + (a_1 + (a_2 + (a_3 + a_4 r_0) r_0) r_0) r_0$$

Unraveling this factorization we get

$$\begin{aligned}
\alpha_4 &= a_4 \\
\alpha_3 &= a_3 + \alpha_4 r_0 \\
\alpha_2 &= a_2 + \alpha_3 r_0 \\
\alpha_1 &= a_1 + \alpha_2 r_0 \\
\alpha_0 &= a_0 + \alpha_1 r_0
\end{aligned}$$

Where $P(r_0) = \alpha_0$.

We can generalize this method to an nth-degree polynomial and write the following algorithm:

Algorithm horner_$P(r_0)$
    $\alpha = a_n$
    for $i = n - 1$ downto 0
        $\alpha = \alpha * r_0 + a_i$
    return α

This algorithm performs an addition, a product, and an assignment in each iteration. Since the total number of iterations is n, then the total number of operation performed by horner_P is

$$T(n) = 2n \in \Theta(n)$$

Obviously this algorithm is much better than computing (1).

1.1 Synthetic division

An important observation is that Horner's method (given above) is the same as the synthetic division of $P(x)$ divided by $x - r_0$, which can easily be verified with the following 4-degree polynomial.

r_0	a_4	a_3	a_2	a_1	a_0
		$\alpha_4 r_0$	$\alpha_3 r_0$	$\alpha_2 r_0$	$\alpha_1 r_0$
	$\alpha_4 = a_4$	$\alpha_3 = a_3 + \alpha_4 r_0$	$\alpha_2 = a_2 + \alpha_3 r_0$	$\alpha_1 = a_2 + \alpha_2 r_0$	$\alpha_0 = a_0 + \alpha_1 r_0$

The proof of this fact follows directly form the Polynomial Remainder Theorem: *given two polynomials $P(x)$ and $Q(x)$, there exists a quotient $C(x)$ and a remainder R such that*

$$P(x) = C(x)Q(x) + R$$

In our case $Q(x) = x - r_0$, therefore

$$P(x) = C(x)(x - r_0) + R \qquad (3)$$

and if $x = r_0$ we get $P(r_0) = R$

We can now show how we can use (3) to compute $P'(r_0)$, the derivative of $P(x)$ in r_0 . First we compute the derivative of (3) keeping in mind that $R = P(r_0)$ is a constant:

$$P'(x) = C(x) + (x - r_0)C'(x)$$

and if $x = r_0$ we get $P'(r_0) = C(r_0)$. So we can use synthetic division on the quotient to obtain $P'(r_0)$.

Since in the synthetic division that we originally used to find $P(r_0)$, the bottom row corresponds to the coefficients of the quotient, we can continue to the next row down with another synthetic division to obtain $P'(r_0)$. This process is illustrated in the following example using a 4th-degree polynomial.

r_0	a_4	a_3	a_2	a_1	a_0
	c_3	c_2	c_1	c_0	$P(r_0)$
	b_2	b_1	b_0	$P'_0(r_0)$	

$c_3, c_2, \ldots, c_0$ represents the quotient and we can continue to the next row with a new synthetic division to obtain $P'(r_0)$.

Complete Horner's Algorithm

We can now complete the algorithm given above so that it also computes $P'(x_0)$. Notice that we compute first $P'(x_0)$ given by the last value of β and then we perform one last computation to find α that corresponds to $P(x_0)$

Algorithm Horner
    INPUT: $a_0, a_1, \ldots, a_n, \;\; x_0$
    $\alpha = a_n$
    $\beta = a_n$
    for $i = n - 1$ downto 1
        $\alpha = \alpha * x_0 + a_i$
        if $i > 1$
            $\beta = \beta * x_0 + \alpha$
    return (α, β)

Example
Find $P(2)$ y $P'(2)$ si $P(x) = x^4 - 2x^3 - 10x^2 + x + 4$

2	1	-2	-10	1	4
		2	0	-20	-38
	1	0	-10	-19	**-34**
		2	4	-12	
	1	2	-6	**-31**	

We get that $P(2) = -34$ and $P'(2) = -31$

1.2 Newton's Method using Horner

From what we said before, we can use Horner's method to efficiently compute $P(r_0)$ and $P'(r_0)$ of a given nth-degree polynomial $P(x)$. Therefore we can use it in conjunction with Newton's Method:

$$r_1 = r_0 - \frac{P(r_0)}{P'(r_0)}$$

In each iteration of Newton's Method we compute $P(r_0)$ and $P'(r_0)$ using Horner's synthetic division method, which requires $\Theta(n)$ operations.

Example
Find one root of $P(x) = x^4 - 2x^3 - 10x^2 + x + 4$ with $\epsilon = 10^{-5}$ starting form $r_0 = 2$.
Iteration 1:
We compute $P(2)$ y $P'(2)$ using Horner's Method (see the previous example), so we get

$$r_1 = 2 - \frac{-34}{-31} = 0.903226$$

Iteration 2:
Now we compute $P(0.903226)$ and $P'(0.903226)$ using Horner's Method:

0.903226	1	-2	-10	1	4
		0.903226	-0.990635	-9.927027	-8.063123
	1	-1.096774	-10.990635	-8.927027	-4.063123
		0.903226	-0.174818	-10.084927	
	1	-0.193548	-11.165452	-19.011954	

so we have that

$$r_1 = 0.903226 - \frac{-4.063123}{-19.011954} = 0.689512$$

and the error $= |r_1 - r_0| = 0.213714 > \epsilon$
Iteration 3:
Now we compute $P(0.689512)$ and $P'(0.689512)$ using Horner's Method:

0.689512	1	-2	-10	1	4
		0.689512	-0.903597	-7.518161	-4.49435
	1	-1.310488	-10.903597	-6.518161	-0.49435
		0.689512	-0.42817	-7.81339	
	1	-0.620976	-11.331768	-14.331551	

so we have that

$$r_1 = 0.689512 - \frac{-0.49435}{-14.331551} = 0.655018$$

and the error$= |r_1 - r_0| = 0.034494 > \epsilon$
Iteration 4:
Now we compute $P(0.655018)$ and $P'(0.655018)$ using Horner's Method:

0.655018	1	-2	-10	1	4
		0.655018	-0.880987	-7.127243	-4.013454
	1	-1.344982	-10.880987	-6.127243	-0.013454
		0.655018	-0.451939	-7.423271	
	1	-0.689964	-11.332926	-13.550513	

so we have that

$$r_1 = 0.655018 - \frac{-0.013454}{-13.527959} = 0.654025$$

and the error $= |r_1 - r_0| = 9.9 \times 10^{-4} > \epsilon$
Iteration 5:
Now we compute $P(0.654025)$ and $P'(0.654025)$ using Horner's Method:

0.654025	1	-2	-10	1	4
		0.654025	-0.880301	-7.115989	-4.00001
	1	-1.345975	-10.880301	-6.115989	-0.00001
		0.654025	-0.452553	-7.41197	
	1	-0.69195	-11.332854	-13.527959	

and we get

$$r_1 = 0.654025 - \frac{-0.00001}{-13.527959} = 0.654024$$

finally, the error $= |r_1 - r_0| = 8.3 \times 10^{-7} < \epsilon$
Therefore, the solution within the required error tolerance is $r = 0.654024$

1.3 Polynomial Deflation

Assume that we have been able to find **one** root r of $P(x)$, then we have that

$$P(r) = 0$$

From the *Polynomial Remainder Theorem* we know that

$$P(x) = C(x)(x - r)$$

since the remainder is $R = 0$ when we divide $P(x)$ by $(x - r)$ and r is a root. Therefore, the equation that we want to solve

$$P(x) = 0$$

implies that

$$C(x)(x - r) = 0$$

from where we can see that either $x - r$=0, i.e. r is a root. Or $C(x) = 0$.

Therefore, after finding one root r of $P(x)$ the remaining roots of $P(x)$ can be found by solving the equation $C(x) = 0$.

Notice that since $C(x)$ is the quotient of dividing an n-th degree polynomial $P(x)$ by a first degree polynomial $x-r$, then $C(x)$ must be a polynomial with degree at most $n-1$. This is why we say that the polynomial has been deflated, because the problem of finding roots of a polynomial of degree n is now reduced to finding the roots of a polynomial of degree $n - 1$.

Notice also that the coefficients of $C(x)$ are precisely the last β's found when using Horner's Method on $P(x)$.

Using this strategy we can write the following "top level" algorithm to find all the roots of a polynomial $P(x)$

```
FindAllRoots(P(x),tolerance)
   repeat
      find an initial point x0 (random?)
      r=NewtonHorner(P(x),x0,tolerance) // find a root r of P(x)
      // C(x) is given by the coefficients in the last step of Horner
      P(x)=C(x)
   until degree(P(x))<=2
   solve P(x) using the quadratic formula
```

Example

Find **all** the roots of $P(x) = x^3 - 4x^2 + x + 6$ with $\epsilon = 10^{-5}$. Start with $x_0 = 1$

```
Iteration 1:

x0=1

1  |  1   -4   1   6
   |       1  -3  -2
   -------------------
   |  1   -3  -2   4
   |       1  -2
   ------------------
      1   -2  -4

x1 = x0 - P(x0)/P'(x0) = 1 - 4/(-4) = 2
error = |x1-x0| = |2-1| = 1 > epsilon

Iteration 1:
x0 = 2

2  |  1   -4   1   6
   |       2  -4  -6
   -------------------
   |  1   -2  -3   0
   |       2   0
   ------------------
      1    0  -3

x1 = x0 - P(x0)/P'(x0) = 2 - 0/(-3) = 2
error = |x1-x0|=|2-2| = 0 < epsilon

No more iteration needed

So first root:

   r1=2
```

```
The deflated polynomial: C(x) = x^2 -2x -3 = 0

Using quadratic equation:
 r2 = (2+4)/2 = 3
 r3 = (2-4)/2 = -1
```

1.4 In Practice

In general when computing a root r using Newton-Horner, there will be an error $< \epsilon$, and so the deflated polynomial $C(x)$ will inherit that error, since $C(x) = \frac{P(x)}{x-r}$. If we then use $C(x)$ to compute one more root of $P(x)$ it will have a larger error than if we had used Newton-Horner directly on $P(x)$. In practice what is done is to use the roots computed using the `FindAllRoots` method given above as an initial value to compute the actual roots using Newton-Horner on $P(x)$ for each solution.