Departament de Disseny i Programació de Sistemes Electrònics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Low level math libraries

Fixed point iteration and polynomial expansions

Jordi Bonet-Dalmau

March 23, 2020

During the *Embedded Systems* course we will talk about some of the problems related with the use of finite-length arithmetic. Rounding or truncation at some point of an arithmetic operation is necessary to satisfy the wordlength requirements of multipliers, data memory, interfaces... Some of these problems are catastrophic cancellation and error accumulation when using IIR filters or adding a large set of numbers. Binary tree summation, compensated summation or error spectrum shaping (ESS) are different techniques to reduce the numerical impact of these problems avoiding an increase in the wordlength and so a higher computational cost.

Here we will deal with an issue related with the complexity with which some operations in fixed or floating point numbers are executed. Microcontrollers, like the AVR of the Arduino you are used to work, have a few set of arithmetic instructions. When programming in languages like C or Python you use any kind of operation beyond the strictly hardware implemented addition and multiplication. Have you ever wondered how a division, a square root or a trigonometric function is performed? These operations are made thanks to the math libraries available for the AVR. When using a microcontroller with limited computational capacity, speed can be more important than accuracy. Next we will show some of the most common techniques used by these libraries.

1 Newton's method

The Newton's method is used to successively obtain better approximations to the roots of a function $f(z)$: i.e. the values of z that makes $f(z) = 0$. This method uses the knowledge of the value of the function at z_0, $f(z_0)$, and its derivative at this point, $f^{'}(z_0)$, to compute the value z_1 at which the tangent of $f(z)$ at $(z_0, f(z_0))$ intersects with the z-axis:

$$z_1 = z_0 - \frac{f(z_0)}{f'(z_0)}. \tag{1}$$

When $f(z)$ is a linear function the tangent is equal to the function and so z_1 will be the root. If $f(z)$ is not a linear function, then z_1 will a better approximation to the root than z_0, provided that z_0 is close enough to the root. This way, we can compute $f(z_1)$ and its derivative at this point, $f^{'}(z_1)$, to compute z_2 that will be closer to the root than z_1:

$$z_2 = z_1 - \frac{f(z_1)}{f'(z_1)}. \tag{2}$$

A geometrical illustration of how the Newton's method works to find out the roots of $f(z) = z^2 - 2$ is showed in *Figure* 1. Starting at $z_0 = 0.5$, $f(z_0) = -1.75$ and $f^{'}(z_0) = 1$ are used to draw the tangent at $(z_0, f(z_0))$. The point $z_1 = 2.25$ where this tangent intersects with the z-axis could be the root, $f(z_1) = 0$, we are looking for. In fact, as stated previously, it would be the root if $f(z)$ were a linear funtion. As this is not what happens, $f(z_1) = 3.062$ and $f^{'}(z_1) = 4.5$ are used to draw the tangent at $(z_1, f(z_1))$, and find out its intersection with the z-axis: $z_2 = 1.569$. Although z_2 is not a root, $f(z_2) = 0.4632$, it is much closer to the root $\sqrt{2} \simeq 1.4142$ we are looking for than the previous values of z.

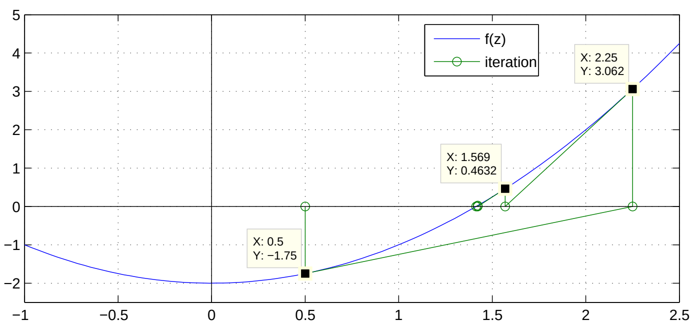

Figure 1: A geometrical illustration of the Newton's method. $f(z) = z^2 - 2$ and $z_0 = 0.5$.

2 Fixed point iteration

The Newton's method is based on iteratively computing

$$z_{n+1} = z_n - \frac{f(z_n)}{f^{'}(z_n)}, \tag{3}$$

starting with z_0 as an initial guess, hoping that the sequence z_n will converge to z, which is a fixed point (here comes the name *fixed point iteration*) of f:

$$z = z - \frac{f(z)}{f^{'}(z)}. \tag{4}$$

Usually we consider that the fixed point is reached when the difference between z_{n+1} and z_n is lower than a selected threshold:

$$\mid z_{n+1} - z_n \mid < \varepsilon. \tag{5}$$

Task 1. Use the Newton's method to find out the roots of $f(z) = z^2 - x$. Consider that the fixed point is reached when the relative difference between z_{n+1} and z_n is lower than a threshold th, i.e.

$$\left| \frac{z_{n+1} - z_n}{z_{n+1}} \right| < th. \tag{6}$$

Note that the absolute error shown in *equation* 5 is well suited for fixed-point arithmetic, while the relative error shown in *equation* 6 is well suited for floating point arithmetic. If we choose *equation* 6 we will have to deal with a discontinuity at $z = 0$, i.e it can not be used if one of the roots of $f(z)$, or any z_n, is (close to) zero.

The following code in Octave is used to find out the roots of $f(z) = z^2 - 2$ with $th = 10^{-6}$.

```
%% computing the roots of f(z)=z^2-x

% parameters
x=2;z0=8;th=1e-6;%th is the threshold

% initialization of variables
Z=[z0]; i=0;

% iteration
do
 i=i+1
 z0=z0/2+x/2/z0
 dz=(z0-Z(end))/z0;
 Z=[Z z0];
 pause
 until abs(dz)<th

% plot of the iteration
n=1:length(Z);
plot(n,Z,'o-',n,ones(1,length(Z))*sqrt(x),'-')
```

Try different values of z_0. What is the difference between $z_0 > 0$ and $z_0 < 0$? What is the difference between $z_0 > \sqrt{2}$ and $0 < z_0 < \sqrt{2}$? And the obvious questions: What happens when $z_0 = 0$? Why? How can it be solved? What happens when $z_0 = \sqrt{2}$?

Task 2. Use the Newton's method to find out the roots of $f(z) = 1/z - x$. First consider $x = 3$ and $z_0 = 0.5$. Next, try different values of z_0. What is the difference between $z_0 > 0$ and $z_0 < 0$? What is the difference between $2/3 > z_0 > 1/3$ and $0 < z_0 < 1/3$? What is the difference between $z_0 > 2/3$ and $0 < z_0 < 2/3$? And the obvious questions: What happens when $z_0 = 0$? What happens when $z_0 = 1/3$? What happens when $z_0 = 2/3$?

Task 3. Find the fixed point iteration that a low-level math library could use to compute the inverse of a number using only addition and multiplication operations. Given a number x, compute $z = 1/x$ following the next steps:

a. Manipulate $z = 1/x$ in order to find out a suitable function $f(z) = 0$.

b. Use the Newton's method to find out the roots of $f(z)$ using *equation* 3.

c. Verify that only addition and multiplication operations are involved in *equation* 3. How many additions and multiplication operations are needed in each iteration?

d. Determine a method to automatically initialize z_0 and start a fixed point iteration to successively compute z_{n+1} until no changes in the 5 more significant digits are observed.

e. As a test, consider $x = 7$ and $z_0 = 1/4$ or $z_0 = 1/8$.

Task 4. Repeat the steps in *Task* 3 to find the fixed point iteration that a low-level math library could use to compute the root of a number using only addition and multiplication operations, i.e. given a number x, compute $z = \sqrt{x}$.

a. If an operation different than addition and multiplication appears in *equation* 3, maybe you could do this operation as an iteration in which only addition and multiplication operations are needed. Hint: if and inversion of z_n is needed to compute each z_{n+1}, you could use results of *Task* 3 to compute $1/z_n$.

b. Determine a method to automatically initialize z_0 and start a fixed point iteration to successively compute z_{n+1} until no changes in the 5 more significant digits are observed.

c. Decide also the accuracy of the inverse computation that appears in each iteration: Hint: You only need to compute $1/z_n$ with high accuracy when you are close to the solution $\sqrt{x}$.

d. As a test, consider $x = 3$ and $z_0 = 1$ or $z_0 = 2$.

3 Polynomial expansion

One particular polynomial expansion is obtained using **Taylor series** to represent a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point.

Given a function $g(z)$, we can compute its value as

$$g(z) = \sum_{n=0}^{\infty} \frac{g^{(n)}(z_0)}{n!}(z - z_0)^n, \tag{7}$$

where $g^n(z_0)$ is the n-th derivative of $g(z)$ evaluated at z_0. The advantage of this series in low math libraries is that any function can be computed using just addition and multiplication operations assuming that the values $g^{(n)}(z_0)$ and $1/n!$ are computed in advance. To immplement this idea we have to truncate the summation that appears in *equation* 7 as

$$g(z) \approx \sum_{n=0}^{N} \frac{g^{(n)}(z_0)}{n!}(z - z_0)^n. \tag{8}$$

The error in this approximation can be bounded by Taylor's theorem.

Let's approximate $\sin(z)$ using *equation* 8 with $z_0 = 0$ and $N = 5$ as

$$sin(z) \approx \sum_{n=0}^{5} \frac{\sin^{(n)}(0)}{n!}(z)^n. \tag{9}$$

The odd derivatives $g^{(n)}(z)$ are $(-1)^{\frac{n-1}{2}} \cos(z)$ and the even derivatives are $(-1)^{\frac{n}{2}} \sin(z)$. The odd derivatives evaluated at $z = z_0$ are $(-1)^{\frac{n-1}{2}}$ and the even derivatives zero. So, *equation* 9 is expressed as

$$sin(z) \approx z - \frac{z^3}{3!} + \frac{z^5}{5!}. \tag{10}$$

The error in this approximation can be bounded to the next term of the Taylor series, i.e. $\frac{z^7}{7!}$.

Task 5. Next consider the sin function and its truncated Taylor series.

a. Argue that the range of z that we must consider to compute any value of $\sin(z)$ is $[-\pi/2, \pi/2)$.

b. Compute the bounded error when $N = 5$.

c. Graphically represent both $\sin(z)$ and its Taylor series approximation when $N = 5$ in the range $[-\pi/2, \pi/2)$.

d. Compute the number of addition and multiplication operations needed to compute the Taylor series approximation considering that $1/n!$ have been previously computed.

e. Repeat the three previous questions when $N = 7$.