Some basic concepts in numerical analysis.

In this course, we are going to study iterative methods for solving matrix equations. We are also going to study finite difference methods for approximating the exact solution of partial differential equations. In these instances, the concepts of consistency, stability and convergence of the numerical methods play an essential role in their understanding. In what follows, we illustrate these basic concepts in the framework of a very simple, but illuminating case, namely, that of finding an approximation x^* to the square root of a given number A by means of the famous Newton's method.

We proceed as follows. First, we assume that we can work with all real numbers and that we can carry out all arithmetic operations exactly. We study the properties of consistency, stability and convergence to devise an algorithm that provides the approximation x^* to $\sqrt{A}$. We then turn to the study of that algorithm in the case we use computers. In this case, we cannot work with all real numbers any more, but a finite number of them. Also, we cannot perform the arithmetic operations exactly. As a consequence rounding errors are introduced in our algorithm. Our ultimate goal

is to find what is the effect of those errors in the approximation x^* by using the concepts of consistency, stability and convergence.

1. Devising an algorithm to compute x^*.

In this section, we devise an algorithm which, in a finite number of steps, provides a number x^* such that

$$|\sqrt{A} - x^*| \leq \epsilon \tag{1.1}$$

for any given positive real number ϵ. To do this, we are going to use Newton's method. this method generates a sequence $\{x^k\}_{k\geq 0}$ as follows

(1.2a) Take x^0 to be a positive number.

(1.2b) Set $x^{k+1} = \frac{1}{2}(x^k + A/x^k)$ for $k \geq 0$.

We are going to show that it is possible to take x^* to be one of the terms of the sequence defined by (1.2) regardless of the value of x^0 and that of ϵ.

1.2 Consistency of Newton's method.

the first thing we must do is to make sure that the method given by (1.2) does have a chance of providing an approximation to $\sqrt{A}$. thus, let us assume that the sequence $\{x^k\}_{k\geq 0}$ converges to the limit x, that is, assume that

$$\lim_{k\to\infty} x^k = x .$$

then, from (1.25), we immediately get that

$$x = \tfrac{1}{2}(x + A/x)$$

and we can deduce that $x^2 = A$. Since by construction all the terms of the sequence $\{x^k\}_{k\geq 0}$ are positive, so is x. this implies that $x = \sqrt{A}$, as wanted.

A more formal, and powerful, way of making sure that we are actually approximating $\sqrt{A}$ and not any other number is by using the concept of consistency. Since we can consider the sequence

(1.3) $$\{x_e^k := \sqrt{A}\}_{k\geq 0}$$

to be the "exact solution", we say that the method given by (1.2) is consistent if it holds for the exact solution.

It is not difficult to verify that (1.2) is indeed consistent since

$$x_e^0 = \sqrt{A} > 0$$

and

$$x_e^{k+1} - \tfrac{1}{2}(x_e^k + A/x_e^k) = \sqrt{A} - \tfrac{1}{2}(\sqrt{A} + A/\sqrt{A}) = 0$$

for all $k \geq 0$.

1.3 A simple property.

Let us now obtain a simple but important property which will be useful in the study of Newton's method (1.2). The property is

(1.4) If $x^0 \neq \sqrt{A}$, then $x^{k+1} > \sqrt{A}$ for all $k \geq 0$.

To prove this, we use the following identity

$$x^{k+1} - \sqrt{A} = \frac{1}{2x^k}(x^k - \sqrt{A})^2 \qquad \text{for } k \geq 0.$$

which follows directly from (1.2b). So, if $k \geq 0$, we have that $x^0 > 0$, by (1.2a), and, since $x^0 \neq \sqrt{A}$, we get that $x^1 > \sqrt{A}$. If we assume that $x^k > \sqrt{A}$, we immediately conclude that $x^{k+1} > \sqrt{A}$ from the above equation. This proves property (1.4).

1.3 Stability with respect to the initial guess.

If a small perturbation on the initial guess x^0 produces small perturbations in all the terms of the sequence generated by Newton's method (1.2), we say that the method is stable with respect to the initial guess.

More precisely, let $\{x_1^k\}_{k\geq 0}$ and $\{x_2^k\}_{k\geq 0}$ be two sequences generated by Newton's method (1.2). We say that it is stable with respect to the initial guess if

$$\sup_{k\geq 1} |x_1^k - x_2^k| \leq C |x_1^0 - x_2^0| \tag{1.5}$$

for some positive constant.

Let us investigate if this is the case. Since, by (1.25)

$$x_1^{k+1} = \frac{1}{2} x_1^k + \frac{A}{2x_1^k} \quad \text{for } k\geq 0,$$
$$x_2^{k+1} = \frac{1}{2} x_2^k + \frac{A}{2x_2^k} \quad \text{for } k\geq 0,$$

we get, for $\delta^k := x_1^k - x_2^k$, that

$$\delta^{k+1} = \frac{1}{2}\delta^k + \frac{A}{2}\left(\frac{1}{x_1^k} - \frac{1}{x_2^k}\right)$$
$$= \frac{1}{2}\delta^k - \frac{A}{2}\,\frac{\delta_k}{x_1^k x_2^k}$$

and so

$$\delta^{k+1} = \frac{1}{2}\left(1 - \frac{A}{x_1^k x_2^k}\right)\delta^k \qquad \text{for } k \geq 0.$$

By the property (1.4), $x_1^k x_2^k > A$ for $k \geq 1$, and so $1 - \frac{A}{x_1^k x_2^k} \in (0,1)$ for $k \geq 1$. This implies that

(1.6a) $$|\delta^{k+1}| \leq \frac{1}{2}|\delta^k| \qquad \text{for } k \geq 1,$$

(1.6b) $$|\delta^1| \leq \frac{1}{2}\left|1 - \frac{A}{x_1^0 x_2^0}\right||\delta^0|$$

As a consequence, we have that

(1.7) $$|\delta^{k+1}| \leq \frac{1}{2^{k+1}}\left|1 - \frac{A}{x_1^0 x_2^0}\right||\delta^0| \qquad \text{for all } k \geq 0,$$

and we immediately see that the stability inequality (1.5) holds with

$$C = \frac{1}{2}\left|1 - \frac{A}{x_1^0 x_2^0}\right|.$$

We have thus proved that Newton's method (1.2) is stable with respect to the initial data.

Note that we have actually proven a much <u>stronger stability result</u>, namely, that, <u>for any $N \geq 1$</u>, we have

(1.8a) $$\sup_{k \geq N} |x_1^k - x_2^k| \leq C_N |x_1^0 - x_2^0|,$$

where

(1.8b) $$C_N = \frac{1}{2^N}\left|1 - \frac{A}{x_0^1 x_0^2}\right|.$$

1.4 Convergence.

Note that the stronger stability result (1.7') means that the effect that a perturbation in the initial guess has on the k-th terms of the sequence generated by Newton's method (1.2) diminishes as k increases.

This stability result, together with the fact the Newton's method is consistent, allows us to prove that the method converges. Indeed, by consistency, we can take $\{x_1^k\}_{k\geq 0}$ to be the "exact solution" sequence, see (1.3). Then, dropping the subindex "2" from x_2^k, the stronger stability estimate (1.8) reads

$$\sup_{k\geq N} |\sqrt{A} - x^k| \leq \frac{1}{2^N} \left| \frac{x^{0^2} - A}{x^0} \right| \quad \text{for all } k \geq 0, \tag{1.9}$$

and we immediately have that the method converges,

$$\lim_{k\to\infty} |\sqrt{A} - x^k| = 0$$

Let us emphasize that we have deduced convergence by using consistency and stability. This relation between these three concepts holds for many algorithms and is important to keep it in mind when studying them.

1.5 An algorithm to compute x^*.

So far, we have only studied the properties of consistency, stability and convergence of Newton's method (1.2). We are now going to use the best outcome of our study, namely, inequality (1.8), to devise an algorithm to actually compute our approximation x^*.

Since by (1.9),

$$|\sqrt{A} - x^N| \leq \frac{1}{2^N} \left| \frac{x^{0^2} - A}{x^0} \right|$$

if we take N such that

$$\frac{1}{2^N} \left| \frac{x^{0^2} - A}{x^0} \right| \leq \epsilon,$$

we can be sure that $|\sqrt{A} - x^N| \leq \epsilon$, as required. Hence, since

$$2^N \geq \frac{\left| \frac{x^{0^2} - A}{x^0} \right|}{\epsilon}$$

we have that

$$N \geq \frac{\log\left(\frac{\left| \frac{x^{0^2} - A}{x^0} \right|}{\epsilon} \right)}{\log 2}.$$

thus, the following algorithm gives us the approximation x^* we sought:

(1.70)

① Pick an initial guess $x^0 > 0$.

② Set N to be the smallest integer satisfying

$$N \geq \frac{\log\left(\left|\frac{x^{0^2} - A}{x^0}\right| / \epsilon\right)}{\log 2}$$

③ For $k = 0, \ldots, N-1$ compute

$$x^{k+1} = \frac{1}{2}\left(x^k + \frac{A}{x^k}\right)$$

④ Set $x^* = x^N$ and stop.

Note that to compute x^{k+1} from x^k we need two divisions and a sum, that is 3 "operations". Thus, the number of operations needed to compute our approximation x^*, also called the computational complexity of the algorithm, is $3N$!

Next, let us get some idea of how big N has to be for a simple case. If we take $A = 5$, $x^0 = 1$ and $\epsilon = 10^{-6}$, we get that

$$N \geq \frac{\log\left(4 \cdot 10^6\right)}{\log 2} \approx 20.$$

1.6 A refinement of the algorithm.

Although the computation of 20 terms of the sequence $\{x^k\}_{k\geq 0}$ does not involve too much computational effort, we can still improve our algorithm to make it faster. We do that by modifying its stopping criterion and rewriting it in terms of the size of the difference $x^{k+1} - x^k$.

To do this, note that by (1.25),

$$x^{k+1} - x^k = -\frac{x^k}{2} + \frac{A}{2x^k}$$

$$= \frac{1}{2x^k}(A - x^{k^2})$$

$$= \left(\frac{\sqrt{A} + x^k}{2x^k}\right)(\sqrt{A} - x^k)$$

and so

$$\sqrt{A} - x^k = \left(\frac{2x^k}{\sqrt{A} + x^k}\right)(x^{k+1} - x^k).$$

By property (1.4), this implies that

$$|\sqrt{A} - x^k| \leq 2\,|x^{k+1} - x^k| \qquad \text{for } k \geq 0,$$

and since, by (1.6a) with $\delta^k := \sqrt{A} - x^k$, we have

$$|\sqrt{A} - x^{k+1}| \leq \frac{1}{2} |\sqrt{A} - x^k| \qquad \text{for } k \geq 1.$$

Hence

$$|\sqrt{A} - x^{k+1}| \leq |x^{k+1} - x^k| \qquad \text{for } k \geq 1$$

We can thus modify our algorithm (1.10) as follows:

(1.11)

① Set $x^0 = \max\{1, A\}$.

② Set N to be the smallest integer such that

$$N \geq \frac{\log\left(\left|\frac{x^{0^2} - A}{x_0}\right| / \epsilon\right)}{\log 2}.$$

③ For $k = 0, \ldots, N-1$

- Set $x^{k+1} = \frac{1}{2}\left(x^k + \frac{A}{x^k}\right)$
- If $k \geq 1$ and $|x^{k+1} - x^k| \leq \epsilon$ go to ④

④ Set $x^* = x^{k+1}$ and stop.

This algorithm provides a possibly different approximation than algorithm (1.10). However, it is potentially faster.

2. Real numbers, machine numbers and rounding.*

When we implement our algorithm (1.11) in any computer a problem arises from the fact that computers cannot work with all real numbers $\mathbb{R}$ but only with a finite subset of them. As a consequence, when "stored" in a computer, an arbitrary real number is usually rounded off. This generates a loss of precision whose effect in the approximation has to be carefully studied. Before doing that, here we explore how real numbers are stored in a computer and how the rounding takes place.

2.1 Real numbers $\mathbb{R}$.

Since any computer uses only ones and zeroes (stored in memory units called "bits") to represent numbers, we begin by recalling how to represent all real numbers $\mathbb{R}$ with only ones and zeroes.

Any number $x \in \mathbb{R}$ can be expressed as

$$x = \pm \left(d_n 2^n + \cdots + d_0 + d_{-1} 2^{-1} + \cdots \right)$$

for some integer n and some $d_i \in \{0,1\}$ for $i \leq n$. We simply write

$$x = \pm \left(d_n d_{n-1} \cdots d_0 \cdot d_{-1} d_{-2} \cdots \right)_2 .$$

* From the book "Numerical Analysis. An Introduction", by W. GAUTSCHI.

Let us give some simple examples:

$$(10.01)_2 = 1\cdot 2^{1} + 0\cdot 2^{0} + 0\cdot 2^{-1} + 1\cdot 2^{-2} = (2.25)_{10}$$

$$(.\overline{01})_2 = 0\cdot 2^{-1} + 1\cdot 2^{-2} + 0\cdot 2^{-3} + 1\cdot 2^{-4} + \cdots$$

$$= 2^{-2} + 2^{-4} + 2^{-6} + \cdots$$

$$= \sum_{m=1}^{\infty} 2^{-2m}$$

$$= \frac{1}{3}$$

$$= (0.\overline{3})_{10}$$

$$(0.0\overline{0011})_2 = (0.1)_{10}$$

$$(0.\overline{0011})_2 = (0.2)_{10}$$

Yet another way of writing all real numbers is the following:

$$x = f \cdot 2^{e}$$

where "f" is the fractional part of x, and "e" is the exponent. the fractional part is of the form

$$f = \pm (.\, b_{-1}\, b_{-2} \cdots)_2$$

and the exponent

$$e = \pm (c_{s-1}\, c_{s-2} \cdots c_0.)_2$$

for some integer s and $b_{-i}, c_j \in \{0,1\}$ for $i = 1,2,\cdots$ and $j = 0,\ldots,s-1$. We assume $b_{-1} \neq 0$!

2.2. The machine numbers $\mathbb{R}(s,t)$.

In a computer, we can only use a finite number of "bits", each containing a "1" or a "0", to represent numbers. So, the machine numbers are those of the form

$$x = f \cdot 2^e$$

where the fractional part is

$$f = \pm (.b_{-1}\, b_{-2} \cdots b_{-t})_2$$

for a fixed number of binary digits "t", and where the exponent is

$$e = \pm (c_{s-1}\, c_{s-2} \cdots c_0)_2$$

where, unlike the real numbers, "s" is a fixed number. This set of numbers is sometimes denoted by $\mathbb{R}(s,t)$. In the computer, each number in $\mathbb{R}(s,t)$ is stored as follows:

$\pm$	b_{-1}	b_{-2}	$\cdots$	b_{-t}	$\pm$	c_{s-1}	$\cdots$	c_0

where it is often assumed that $b_{-1} = 1$. In this case, the number is called "normalized".

It is not difficult to see that

(2.1a) $$\max_{x \in \mathbb{R}(s,t)} |x| = (.\underbrace{111\ldots1}_{t \text{ times}})\, 2^{\overbrace{(111\ldots1.)}^{s \text{ times}}}$$

$$= (1 - 2^{-t}) \cdot 2^{(2^s - 1)} =: M$$

and that

(2.1b) $$\min_{x \in \mathbb{R}(s,t)} |x| = (.\underbrace{100\ldots0}_{t-1 \text{ times}})\, 2^{-\overbrace{(111\ldots1.)}^{s \text{ times}}}$$

$$= \frac{1}{2}\, 2^{-(2^s - 1)}$$

$$= 2^{-2^s} =: m$$

Now, given any $x \in \mathbb{R}$ such that

(2.2a) $$x = \pm (.b_{-1} b_{-2} \ldots)_2\, 2^{\pm (c_{s-1} \ldots c_0)_2}$$

such that $|x| \in [m, M]$, we say that $\underline{x} \in \mathbb{R}(s,t)$ is obtained from x by <u>chopping</u> if

(2.2b) $$\underline{x} = \pm (.b_{-1} b_{-2} \ldots b_{-t})\, 2^{\pm (c_{s-1} \ldots c_0)_2}$$

We then write $\underline{x} = \mathrm{chop}(x)$.

Note that, if we set $e = \pm(c_{s-1} \cdots c_0)_2$,

$$x - \mathrm{chop}(x) = \pm \left(\sum_{k=t+1}^{\infty} b_{-k} 2^{-k} \right) \cdot 2^e$$

and so

$$|x - \mathrm{chop}(x)| \leq \sum_{k=t+1}^{\infty} 2^{-k} \cdot 2^e$$

$$\leq 2^{-t} \cdot 2^e.$$

Moreover, since

$$|x| \geq \frac{1}{2} \cdot 2^e,$$

we obtain that

$$\frac{|x - \mathrm{chop}(x)|}{|x|} \leq 2 \cdot 2^{-t}$$

A better approximation of $x \in \mathbb{R}$ by elements of $\mathbb{R}(s,t)$ is given by the so-called <u>symmetric rounding</u>, $\mathrm{rd}(x)$. It is related to chopping by the following simple relation

$$\mathrm{rd}(x) = \mathrm{chop}\left(x + \tfrac{1}{2} 2^{-t} \cdot 2^e\right) \tag{2.3}$$

It is not difficult to show that

$$\frac{|x - rd(x)|}{|x|} \leq 2^{-t}.$$

Hence

$$rd(x) = x(1+\varepsilon) \qquad \text{where } |\varepsilon| \leq 2^{-t}.$$

↑ "machine precision"

2.3 Machine arithmetic.

Due to rounding, if we perform any arithmetic operation "$\circ$" (where "$\circ$" represents "$+$", "$-$", "$\times$", or "$\div$") between two arbitrary real numbers, the result in the computer is not $x \circ y$ but

$$(x \circ y)(1+\varepsilon) \qquad \text{where } |\varepsilon| < 2^{-t}. \tag{2.4}$$

Let us explore what happens if we provide $x, y \in \mathbb{R}$ to the computer and we perform a multiplication. By simply storing x and y in the computer, these numbers are distorted by rounding; this results in the fact that instead of x and y, we will be working with $x(1+\varepsilon_x)$ and $y(1+\varepsilon_y)$, where $|\varepsilon_x|$ and $|\varepsilon_y|$ are bounded by 2^{-t}. When we multiply these two numbers we get, by (2.4)

$$x(1+\varepsilon_x)\, y(1+\varepsilon_y)(1+\varepsilon)$$

and we see that the relative error satisfies

$$\frac{|xy - x(1+\epsilon_x)\, y(1+\epsilon_y)(1+\epsilon)|}{|xy|} \leq |1-(1+\epsilon_x)(1+\epsilon_y)(1+\epsilon)|$$

$$\leq (1+2^{-t})^3 - 1$$

$$= 3\,2^{-t} + 3\,2^{-2t} + 2^{-3t}$$

$$\approx 3 \cdot 2^{-t}.$$

In other words, the relative error is of the order of the machine precision. This also happens with the division.

Let us now see the case of addition. Proceding as before, we obtain that the result is

$$(x(1+\epsilon_x) + y(1+\epsilon_y))(1+\epsilon).$$

Hence, the relative error satisfies

$$\frac{|(x+y) - (x(1+\epsilon_x) + y(1+\epsilon_y))(1+\epsilon)|}{|x+y|} \leq \left|(1+\epsilon)\left(1 + \frac{x\epsilon_x + y\epsilon_y}{x+y}\right) - 1\right|$$

$$\leq (1+2^{-t})\left(1 + \left|\frac{x\epsilon_x + y\epsilon_y}{x+y}\right|\right) - 1$$

Now, if x and y are of the same sign, we obtain

$$\left|\frac{x\epsilon_x + y\epsilon_y}{x+y}\right| \leq \left(\left|\frac{x}{x+y}\right| + \left|\frac{y}{x+y}\right|\right) 2^{-t} = 2^{-t}$$

and the relative error is not bigger than

$$(1-2^{-t})^2 - 1 \approx 2 \cdot 2^{-t}.$$

However, if x and y have opposite sides, the relative error can be huge. For example, if we take

$$y = -x + 2^{-t}, \qquad x = 1,$$

we have that $\left|\frac{x}{x+y}\right| = 2^t$ and $\left|\frac{y}{x+y}\right| = 2^t - 1$, and so

$$\left|\frac{x\,\epsilon_x + y\,\epsilon_y}{x+y}\right| \leq 2$$

and the relative error is bounded by

$$(1+2^{-t})(1+2) - 1 \approx 2!$$

Thus, it is this kind of operations that must be avoided in scientific computing. A similar phenomenon happens with subtraction.

3 Working with the computer.

We are now ready to study algorithm (1.11) as implemented in a computer. We begin by studying Newton's method, just as we did in the first section. The difference being that now we are going to be working with the machine numbers $\mathbb{R}(s,t)$ instead of the whole real numbers $\mathbb{R}$. We are going to pay careful attention to the introduction of rounding errors and their "propagation" by the algorithm. In particular, we will see that thanks to the consistency and stability properties of Newton's methods, the rounding errors are "controlled" and do not spoil the quality of the approximation x^* given by algorithm (1.11).

3.1. Newton's method in $\mathbb{R}(s,t)$.

Newton's method (1.2) reads as follows when working with $\mathbb{R}(s,t)$:

(3.1a) Take x^0 to be a positive number in $\mathbb{R}(s,t)$.

(3.1b) Set

$$x^{k+1} = \frac{1}{2}\left(x^k + \frac{A(1+\varepsilon_A)}{x^k}(1+\varepsilon_1^k)\right)(1+\varepsilon_2^k)(1+\varepsilon_3^k)$$

for all $k \geq 0$.

Here, we have that

(3.1c) $$|\varepsilon_A|,\ |\varepsilon_i^k| \leq 2^{-t} \quad \text{for } i=1,2,3 \text{ and all } k \geq 0.$$

Let us justify this form of the method. Let us begin by noting that since A is a real number, when stored by the computer, it has to be rounded off. As a consequence A is replaced by $A(1+\epsilon_A)$, a number in $\mathbb{R}(s,t)$. The numbers ϵ_1^k, ϵ_2^k and ϵ_3^k appear due to the arithmetic operations. So ϵ_1^k appear due to the division of $A(1+\epsilon_A)$ by x^k (see (2.4)!), ϵ_2^k due to the addition of the resulting number and x^k, and ϵ_k^3 due to the division by 2.

Next, let us note that we can rewrite (3.1b) as follows:

$$x^{k+1} = \frac{1}{2}(x^k + A/x^k) + R^k \tag{3.2a}$$

where

$$R^k = \frac{x^k}{2}\left((1+\epsilon_2^k)(1+\epsilon_2^k)-1\right) + \frac{A}{2x^k}\left((1+\epsilon_A)(1+\epsilon_1^k)(1+\epsilon_2^k)(1+\epsilon_3^k)-1\right). \tag{3.2b}$$

We can immediately see that if we were working with infinite precision, that is, if $t=\infty$, $\epsilon_A = \epsilon_1^k = \epsilon_2^k = \epsilon_3^k = 0$ and $R^k = 0$. We would recover then the original equation (1.25), as expected. In other words, the term R^k introduced by rounding is small. We must understand how small R^k is and how it affects the terms of the sequence $\{x^k\}_{k\geq 0}$ generated by (3.1).

3.2 A simple property.

Next, we obtain a property similar to (1.4). It is the following:

(3.3) $$x^{k+1} \geq \sqrt{A}\,(1-\bar{2}^{t})^{3} \qquad \text{for all } k \geq 0.$$

To prove this inequality, let us rewrite (3.1b) as

(3.4a) $$x^{k+1} = \tfrac{1}{2}\,(x^{k} + A^{k}/x^{k})\,(1+\varepsilon^{k})$$

where

(3.4b) $$A^{k} := A\,(1+\varepsilon_{A})\,(1+\varepsilon_{1}^{k}),$$

and

(3.4c) $$(1+\varepsilon^{k}) = (1+\varepsilon_{2}^{k})\,(1+\varepsilon_{3}^{k}).$$

Now, from (3.4a), we obtain

$$x^{k+1} - \sqrt{A^{k}}\,(1+\varepsilon^{k}) = \tfrac{1}{2}\,(x^{k} + A^{k}/x^{k} - \sqrt{A^{k}})\,(1+\varepsilon^{k})$$

$$= \frac{(x^{k} - \sqrt{A^{k}})^{2}}{2x^{k}} \cdot (1+\varepsilon^{k}).$$

Since $x^{0} > 0$, we immediately see that

$$x^{1} \geq \sqrt{A^{1}}\,(1+\varepsilon^{k}) \geq \sqrt{A}\,(1-\bar{2}^{t})^{3},$$

by (3.4b) and (3.4c). Now, assume that $x^{k} \geq \sqrt{A}\,(1-\bar{2}^{t})^{3}$. Then

$$x^{k+1} \geq \sqrt{A^{k}}\,(1+\varepsilon^{k}) \geq \sqrt{A}\,(1-\bar{2}^{t})^{3}.$$

This proves property (3.3).

3.3 Yet another property.

In order to obtain an estimate of the size of the perturbation R^k, we are going to need another property of the method (3.1). It is the following. We have that

$$x^{k+1} \leq \rho^k x^1 + \Theta/(1-\rho) \qquad \text{for all } k \geq 1, \tag{3.5a}$$

where

$$\rho = \tfrac{1}{2}\left(1 + 2^{-t}\right)^2 \tag{3.5b}$$

and

$$\Theta = \tfrac{1}{2}\frac{\left(1+2^{-t}\right)^4}{\left(1-2^{-t}\right)^3}\sqrt{A}, \tag{3.5c}$$

provided

$$t \geq 2. \tag{3.5d}$$

By the equation (3.4a), we have that

$$|x^{k+1}| \leq \frac{|1+\varepsilon^k|}{2}|x^k| + \frac{|1+\varepsilon^k|}{2}\frac{|A^k|}{|x^k|} \quad \text{for } k \geq 0.$$

Since by (3.1a) $x^0 > 0$, and by (3.3) $x^{k+1} > 0$ for $k \geq 0$, we obtain that

$$x^{k+1} \leq \frac{|1+\varepsilon^k|}{2}x^k + \frac{|1+\varepsilon^k|}{2}\frac{|A^k|}{x^k} \qquad \text{for } k \geq 0.$$

By (3.4b) and (3.4c), using the estimates (3.1c),

we get that

$$X^{k+1} \le \frac{(1+\bar{z}^t)^2}{2} X^k + \frac{(1+\bar{z}^t)^4}{2} \frac{A}{x^k} \qquad \text{for } k \ge 0.$$

Finally, by property (3.3) with "k+1" replaced by "k",

$$X^{k+1} \le \frac{(1+\bar{z}^t)^2}{2} X^k + \frac{(1+\bar{z}^t)^4}{2} \cdot \frac{\sqrt{A}}{(1-\bar{z}^t)^3} \qquad \text{for } k \ge 1,$$

or, using (3.5b) and (3.5c),

$$X^{k+1} \le \rho X^k + \theta \qquad \text{for } k \ge 1.$$

This immediately implies that

$$X^{k+1} \le \rho^k X^1 + \sum_{i=0}^{k-1} \rho^i \theta$$

$$= \rho^k X^1 + \left(\frac{1-\rho^k}{1-\rho}\right)\theta \qquad \text{for } k \ge 1.$$

By (3.5d), $\rho < 1$, and $(1-\rho^k)/(1-\rho) \le 1/(1-\rho)$ for all $k \ge 1$. This proves the property (3.5a).

3.4 The size of R^k.

Armed with properties (3.3) and (3.5), we are now ready to estimate the size of the perturbation R^k. By the definition of R^k (3.2b) and the estimate (3.1c), we have that

this implies that we can write

(3.7) $$|R^k| \leq C \cdot 2^{-t} \qquad \text{for all } k \geq 0.$$

In other words, the size of the perturbation is of the order of the machine precision.

3.5 Behavior of the error $\sqrt{A} - x^k$.

Now we investigate the behavior of the error $\sqrt{A} - x^k$ as k increases. We want to see what estimate we can obtain instead of the estimate (1.9).

thus, we carry out an analysis similar to the one leading to estimate (1.9). thus, using (3.2a), we obtain

$$\sqrt{A} - x^{k+1} = \frac{1}{2}\left(1 - \frac{\sqrt{A}}{x^k}\right)\left(\sqrt{A} - x^k\right) - R^k$$

and so

(3.8) $$\left|\sqrt{A} - x^{k+1}\right| \leq \frac{1}{2}\left|1 - \frac{\sqrt{A}}{x^k}\right| \left|\sqrt{A} - x^k\right| + C \cdot 2^{-t},$$

by (3.7). Now, by property (3.3)

$$\frac{1}{2}\left|1 - \frac{\sqrt{A}}{x^k}\right| \leq \frac{1}{2} \max\left\{1, \frac{1}{(1-2^{-t})^3} - 1\right\} =: K$$

for all $k \geq 1$.

If $t \geq 3$, we can take $k = \frac{1}{2}$ and we obtain that

$$|\sqrt{A} - x^{k+1}| \leq \frac{1}{2} |\sqrt{A} - x^k| + c \cdot 2^{-t} \quad \text{for all } k \geq 1.$$

this immediately implies that

(*)
$$\begin{cases} |\sqrt{A} - x^{k+1}| \leq \frac{1}{2^k} |\sqrt{A} - x^1| + \boxed{\sum_{i=0}^{k-1} \left(\frac{1}{2}\right)^i \cdot c \cdot 2^{-t}} \\ \qquad = \frac{1}{2^k} |\sqrt{A} - x^1| + \boxed{\frac{1 - \left(\frac{1}{2}\right)^k}{1 - \frac{1}{2}} \cdot c \cdot 2^{-t}} \\ \qquad \leq \frac{1}{2^k} |\sqrt{A} - x^1| + \boxed{2 \cdot c \cdot 2^{-t}} \end{cases} \quad \text{for } k \geq 1.$$

↙ effect of rounding

This inequality is trivially true for $k = 0$, and since

$$|\sqrt{A} - \hat{x}^1| \leq \frac{1}{2} \left| \frac{x^{0^2} - A}{x^0} \right| + c \cdot 2^{-t}$$

by (3.8) with $k = 0$, we obtain

(3.9)
$$\sup_{k \geq N} |\sqrt{A} - x^k| \leq \frac{1}{2^N} \left| \frac{x^{0^2} - A}{x^0} \right| + 3 \cdot c \cdot 2^{-t} \quad \text{for all } k \geq 0$$

Thus, comparing this with the estimate (1.9), we see that the only effect of the rounding is in the term

$$3 \cdot c \cdot 2^{-t}$$

which is of the order of machine precision! Reviewing the calculations leading to (3.9), we can see the effect of the consistency and the stability of Newton's method. they allow us to "control" the effect of the perturbation terms R^k; see the boxed terms in the sequence of inequalities in (*)

As a consequence, the approximation x^* given by the algorithm (1.11) satisfies the estimate

$$|\sqrt{A} - x^*| \leq \epsilon + 3 \cdot C \cdot \bar{2}^r$$

In other words, the approximation x^* satisfies our accuracy requirement "up to machine precision".

Let us end by pointing out that the parameter ϵ cannot be too small. (Why?).