Numerical Computing: Introduction

Fengyan Li

Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY

Spring 2019

- For many applications in sciences and engineering, one can set up mathematical models in order to understand them.
- Very rarely, these models can be solved analytically. Instead one can reply on computational / numerical methods to provide approximate solutions.
- Designing an accurate robust computational algorithm with good cost efficiency can be non-trivial. Fortunately, this task often can be divided into a set of simpler problems.
- In this course, *we will examine some basic problems in scientific computing, such as root finding, interpolation, solving initial value problems, and examine how to solve them numerically.*

Quote from Donald Knuth [1997], the creator of TeX, regarding the challenges of finite precision arithmetic:

"We don't know how much of the computer's answers to believe. Novice computer users solve this problem by implicitly trusting in the computer as an infallible authority; they tend to believe that all digits of a printed answer are significant. Disillusioned computer users have just the opposite approach; they are constantly afraid that their answers are almost meaningless.

Every well-rounded programmer ought to have a knowledge of what goes on during the elementary steps of floating point arithmetic. This subject is not at all as trivial as most people think, and it involves a surprising amount of interesting information."

Examples (motivation to study finite precision arithmetic)

Example 1: Compute the partial sum of the harmonic series[1]

$$\sum_{k=1}^{n} \frac{1}{k}$$

- Algorithm 1: to add from the largest to the smallest term

$$S(n) = 1 + \frac{1}{2} + \cdots + \frac{1}{n-1} + \frac{1}{n}$$

- Algorithm 2: to add from the smallest to the largest term

$$s(n) = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1$$

[1] a divergent series

The difference $S(n) - s(n)$ for $n = 10^j$, $j = 1, 2, 3, 4, 5, 6$.
Note: E-16=10^{-16}

n	$S(n) - s(n)$
10^1	0
10^2	-8.88E-16
10^3	2.66E-15
10^4	-3.73E-14
10^5	-7.28E-14
10^6	-7.83E-13

Discussions:

- Mathematically, $S(n) = s(n)$; computationally, they are not.
- The difference grows with n.
- Question: which approximation is more accurate in general?

The difference $S(n) - s(n)$ for $n = 10^j$, $j = 1, 2, 3, 4, 5, 6$.
Note: E-16=10^{-16}

n	$S(n) - s(n)$
10^1	0
10^2	-8.88E-16
10^3	2.66E-15
10^4	-3.73E-14
10^5	-7.28E-14
10^6	-7.83E-13

Discussions:

- Mathematically, $S(n) = s(n)$; computationally, they are not.
- The difference grows with n.
- Question: which approximation is more accurate in general?

The difference $S(n) - s(n)$ for $n = 10^j$, $j = 1, 2, 3, 4, 5, 6$.
Note: E-16=10^{-16}

n	$S(n) - s(n)$
10^1	0
10^2	-8.88E-16
10^3	2.66E-15
10^4	-3.73E-14
10^5	-7.28E-14
10^6	-7.83E-13

Discussions:

- Mathematically, $S(n) = s(n)$; computationally, they are not.
- The difference grows with n.
- Question: which approximation is more accurate in general?

Example 2:

- Let $f(x) = \sqrt{x}$ for $x > 0$, and we know $f'(x) = \frac{1}{2\sqrt{x}}$.
- Define a function

$$y(k) = \frac{f(16+k) - f(16)}{k} = \frac{\sqrt{16+k} - 4}{k}, \qquad (1)$$

then

$$\lim_{k \to 0} y(k) = f'(16) = \frac{1}{8}. \qquad (2)$$

Compute and plot the function $y(k)$ as $k \to 0$.

Example 2:

- Let $f(x) = \sqrt{x}$ for $x > 0$, and we know $f'(x) = \frac{1}{2\sqrt{x}}$.
- Define a function

$$y(k) = \frac{f(16+k) - f(16)}{k} = \frac{\sqrt{16+k} - 4}{k}, \tag{1}$$

then

$$\lim_{k \to 0} y(k) = f'(16) = \frac{1}{8}. \tag{2}$$

Compute and plot the function $y(k)$ as $k \to 0$.

Example 2:

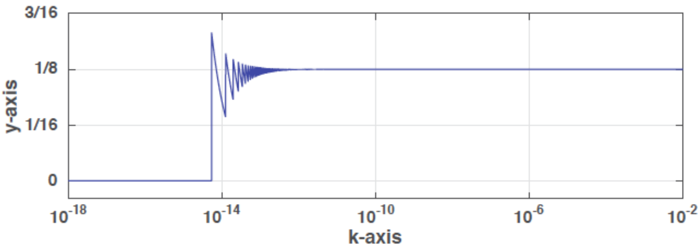
- Let $f(x) = \sqrt{x}$ for $x > 0$, and we know $f'(x) = \frac{1}{2\sqrt{x}}$.
- Define a function

$$y(k) = \frac{f(16+k) - f(16)}{k} = \frac{\sqrt{16+k} - 4}{k}, \tag{1}$$

then

$$\lim_{k \to 0} y(k) = f'(16) = \frac{1}{8}. \tag{2}$$

Compute and plot the function $y(k)$ as $k \to 0$.

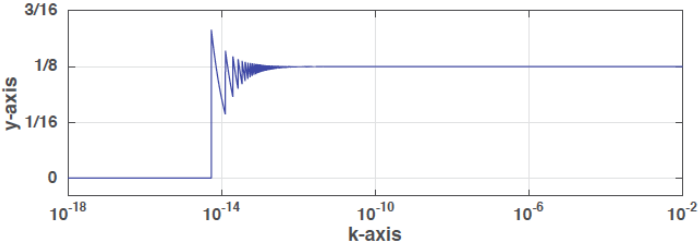


- As k decreases up to $k = 10^{-12}$, $y(k)$ is a good approximation for $f'(16) = \frac{1}{8}$.
- When k further decreases, $y(k)$ starts to oscillates and the errors are visible; After k drops below 10^{-14}, the computed $y(k)$ is around 0.

something seems to be wrong!

- Loss of significance: remedy by reformulation:

$$\frac{\sqrt{16+k}-4}{k} = \frac{1}{\sqrt{16+k}+4}$$

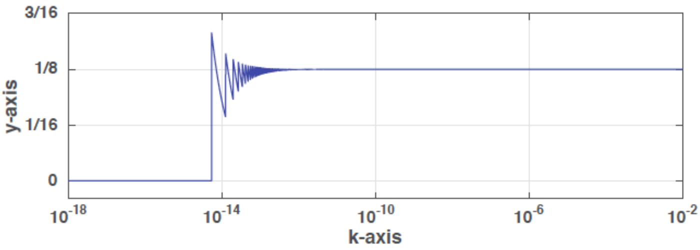


- As k decreases up to $k = 10^{-12}$, $y(k)$ is a good approximation for $f'(16) = \frac{1}{8}$.
- When k further decreases, $y(k)$ starts to oscillates and the errors are visible; After k drops below 10^{-14}, the computed $y(k)$ is around 0.

something seems to be wrong!

- Loss of significance: remedy by reformulation:

$$\frac{\sqrt{16+k}-4}{k} = \frac{1}{\sqrt{16+k}+4}$$

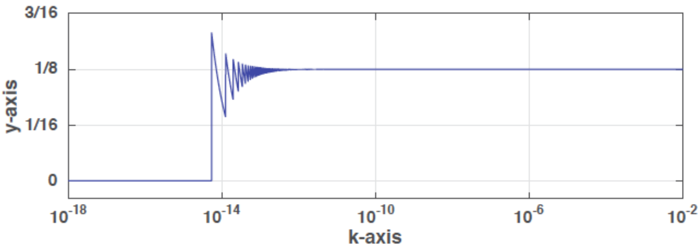


- As k decreases up to $k = 10^{-12}$, $y(k)$ is a good approximation for $f'(16) = \frac{1}{8}$.
- When k further decreases, $y(k)$ starts to oscillates and the errors are visible; After k drops below 10^{-14}, the computed $y(k)$ is around 0.

something seems to be wrong!

- Loss of significance: remedy by reformulation:

$$\frac{\sqrt{16+k}-4}{k} = \frac{1}{\sqrt{16+k}+4}$$

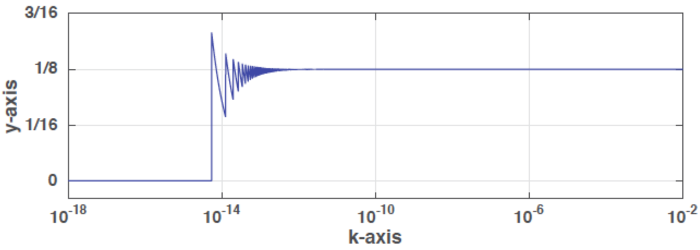


- As k decreases up to $k = 10^{-12}$, $y(k)$ is a good approximation for $f'(16) = \frac{1}{8}$.
- When k further decreases, $y(k)$ starts to oscillates and the errors are visible; After k drops below 10^{-14}, the computed $y(k)$ is around 0.

something seems to be wrong!

- Loss of significance: remedy by reformulation:

$$\frac{\sqrt{16+k}-4}{k} = \frac{1}{\sqrt{16+k}+4}$$



- As k decreases up to $k = 10^{-12}$, $y(k)$ is a good approximation for $f'(16) = \frac{1}{8}$.
- When k further decreases, $y(k)$ starts to oscillates and the errors are visible; After k drops below 10^{-14}, the computed $y(k)$ is around 0.

something seems to be wrong!

- Loss of significance: remedy by reformulation:

$$\frac{\sqrt{16+k}-4}{k} = \frac{1}{\sqrt{16+k}+4}$$

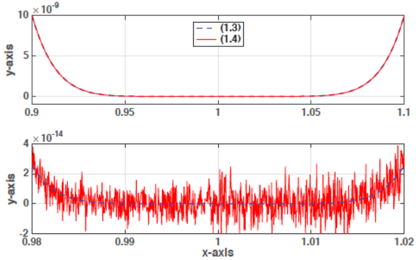
Example 3: Consider the function

$$y(x) = (x-1)^8, \tag{3}$$

and its expanded form

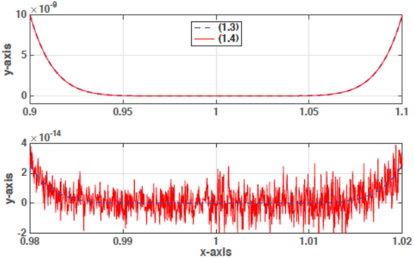
$$y(x) = x^8 - 8x^7 + 28x^6 - 56x^5 + 70x^4 - 56x^3 + 28x^2 - 8x + 1. \tag{4}$$

Evaluate and plot these two functions for $x \in [0.9, 1.1]$. In addition, zoom in the plots for $x \in [0.98, 1.02]$.



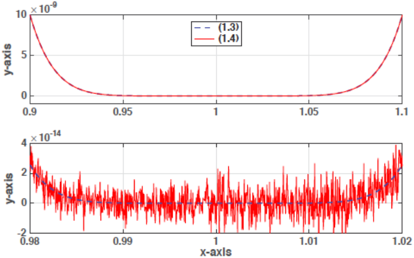
Discussions:

- Again, two mathematically identical functions are not the same computationally.
- The expanded form even leads to negative values.



Discussions:

- Again, two mathematically identical functions are not the same computationally.
- The expanded form even leads to negative values.



Discussions:

- Again, two mathematically identical functions are not the same computationally.
- The expanded form even leads to negative values.

Computational complexity: polynomial evaluation

For example 3, the computational costs can differ greatly. Consider a polynomial $p(x) = 2x^4 + 3x^3 - 3x^3 + 5x - 1$, with all coefficients given and stored (*such as being a vector*).

> **Question:** how many $+, \times$ are needed to evaluate $p(\frac{1}{2})$?

Approach 1: "straightforward"

$$
\begin{array}{rr}
2 \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} & 4(\times) \\
3 \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} & 3(\times) \\
(-3) \cdot \frac{1}{2} \cdot \frac{1}{2} & 2(\times) \\
5 \cdot \frac{1}{2} & 1(\times)
\end{array}
$$

Computational cost: $N_\times = 10, N_+ = 4$

Approach 2: "use more storage"

$$\left(\frac{1}{2}\right)^2 = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \text{ (stored)} \qquad 1(\times)$$

$$\left(\frac{1}{2}\right)^3 = \frac{1}{2} \cdot \left(\frac{1}{2}\right)^2 = \frac{1}{8} \text{ (stored)} \qquad 1(\times)$$

$$\left(\frac{1}{2}\right)^4 = \frac{1}{2} \cdot \left(\frac{1}{2}\right)^3 = \frac{1}{16} \qquad 1(\times)$$

multiplying the coefficients: $4(\times)$.

Computational cost: $N_\times = 7$, $N_+ = 4$

Approach 3: Horner's method based on nested form

$$\begin{aligned} p(x) &= x(2x^3 + 3x^2 - 3x + 5) - 1 \\ &= x(x(2x^2 + 3x - 3) + 5) - 1 \\ &= x(x(x(2 \cdot x + 3) - 3) + 5) - 1 \end{aligned}$$

Computational cost: $N_\times = 4$, $N_+ = 4$

In general

Computational complexity to evaluate a polynomial of degree d

- $N_{add} = d$
- $$N_{mul} = \begin{cases} \sum_{k=1}^{d} k = \frac{(1+d)d}{2} & \text{(approach 1)} \\ 2d - 1 & \text{(approach 2)} \\ d & \text{(approach 3)} \end{cases}$$

Different implementations can mean different cost efficiency!

Listing 1: Example code

```
function y = myPolyEval(x, a, d)
% to evaluate a polynomial at x
%
% inputs: x: where the polynomial p(x) is evaluted
%       d: the degree of p(x)
%       a: a vector of length d+1, that contains
%         the coefficients of p(x), with the constant
%         term p(0) being the last
% output: y= p(x)
%%%%%%%%%%%%%%%%%%

if ((length(a)-d)==1)
   y = a(1);
   for j =2:d+1
      y = y*x+a(j);
   end
else
   disp('error: inconsistency in polynomial degree!')
end
```