Introduction

CS/SE 4X03

Ned Nedialkov

McMaster University

September 10, 2021

Introduction
CS/SE 4X03

Ned Nedialkov

McMaster University

September 10, 2021

Outline

Taylor series

Taylor series of an infinitely differentiable (real or complex) f at c

$$\begin{aligned} f(x) &= f(c) + f'(c)(x-c) + \frac{f''(c)}{2!}(x-c)^2 + \cdots \\ &= \sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!}(x-c)^k \end{aligned}$$

Maclaurin series $c = 0$

$$\begin{aligned} f(x) &= f(0) + f'(c)x + \frac{f''(0)}{2!}x^2 + \cdots \\ &= \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!}x^k \end{aligned}$$

Taylor series cont.

Assume f has $n+1$ continuous derivatives in $[a,b]$, denoted $f \in C^{n+1}[a,b]$
Then for any c and x in $[a,b]$

$$f(x) = \sum_{k}^{n} \frac{f^{(k)}(c)}{k!}(x-c)^k + E_{n+1},$$

where

$$E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1} \quad \text{and } \xi = \xi(c,x) \text{ is between } c \text{ and } x$$

Replacing x by $x+h$ and c by x, we obtain

$$f(x+h) = \sum_{k}^{n} \frac{f^{(k)}(x)}{k!}h^k + E_{n+1},$$

where $E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1}$ and ξ is between x and $x+h$

Taylor series cont.

We say the error term E_{n+1} is of order $n+1$ and write as

$$E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!} h^{n+1} = O(h^{n+1})$$

That is,

$$|E_{n+1}| \leq c h^{n+1}, \quad \text{for some } c > 0$$

Taylor series cont.

Example 1. How to approximate e^x for given x?

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Suppose we approximate using $e^x \approx 1 + x + \frac{x^2}{2!}$
Then

$$e^x = 1 + x + \frac{x^2}{2!} + E_3, \quad \text{where } E_3 \quad = \frac{e^\xi}{3!}x^3, \quad \xi \text{ between } 0 \text{ and } x$$

Let $x = 0.1$. Then $e^{0.1} \approx 1.1052$. The error is

$$E_3 = \frac{e^\xi}{3!}x^3 \lessapprox \frac{1.1052}{3!}0.1^3 \approx 1.8420 \times 10^{-4}$$

Taylor series cont.

How to check our calculation?

Example 2. We can compute a more accurate value using MATLAB's `exp` function.
The error in our approximation is

$$\texttt{exp(x)-(1+x+x\^{}2/2)} \approx 1.7092 \times 10^{-4}$$

This is within the bound 1.8420×10^{-4}:

$$1.7092 \times 10^{-4} < 1.8420 \times 10^{-4}$$

Taylor series cont.

Example 3. If we approximate using three terms

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}$$

the error is

$$E_4 = \frac{e^\xi}{4!}x^4 \lessapprox \frac{1.1052}{4!}0.1^4 \approx 4.6050 \times 10^{-6}$$

Using `exp(0.1)`, the error is

`exp(x)-(1+x+x^2/2+x^3/6)` $\approx 4.2514 \times 10^{-6}$

Errors in computing

Roundoff errors

Example 4.

- Consider computing `exp(0.1)`
- `0.1` binary's representation is infinite:

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

- In floating-point arithmetic, this binary representation is rounded: roundoff error
- The input to the `exp` function is not exactly 0.1 but $0.1 + \epsilon$, for some ϵ
- The input to `exp` is $0.1 + \epsilon$
- The `exp` function has its own error
- Then the output of `exp(0.1)` is rounded when converting from binary to decimal

Errors in computing cont.

Example 5. Compute `(3*(4/3-1)-1)*2^52` in favourite language

exact value	0
double precision	−1
single precision	536870912

Example 6. This code

```
#include <stdio.h>
int main() {
  int    i = 0, j = 0;
  float  f;
  double d;
  for (f = 0.5; f < 1.0; f += 0.1)  i++;
  for (d = 0.5; d < 1.0; d += 0.1)  j++;
  printf("float loop %d  double loop %d \n", i, j);
}
```

outputs `float loop 5 double loop 6`

Errors in computing cont.

Example 7. Let $a_i = i \cdot a_{i-1} - 1$, where $a_0 = e - 1$. Find a_{25}

```c
#include <stdio.h>
#include <math.h>
int main(){
  double a = exp(1.0)-1;
  for (int i = 1; i <= 25; i
     ++)
    a = i * a - 1;
  printf("%e\n", a);
  return 0;
}
```

Matlab

```matlab
a = exp(1.0)-1;
for i = 1:25
    a = i * a - 1;
end
fprintf('%e\n', a);
```

true value	$\approx$ 3.993873e-02	
C	-2.242373e+09	clang v11.0.3, MacOS X
Matlab	4.645988e+09	R2020b
Octave	-2.242373e+09	

Errors in computing cont.

In Matlab, do `doc vpa`

- `vpa(x)`
  - uses variable-precision floating-point arithmetic (VPA)
  - evaluate each element of `x` to $\geq$ `d` significant digits
  - `d` is the value of the `digits` function;
    default default value of digits is 32.
- `vpa(x,d)` uses at least $\geq$ `d` significant digits

Example 7. cont.

```
a = exp(vpa(1))-1;
for i = 1:25
    a = i * a - 1;
end
fprintf('%e \n', a);
```

outputs `3.993873e-02`

Errors in computing cont.

Truncation errors

Consider

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \sum_{k=4}^{\infty} \frac{x^k}{k!}$$

Suppose we approximate

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}$$

That is we truncate the series. The resulting error is truncation error

Errors in computing cont.

Approximating first derivative

$f(x)$ scalar with continuous second derivative

$$\begin{aligned} f(x+h) &= f(x) + f'(x)h + \frac{f''(\xi)}{2}h^2, \quad \xi \text{ between } x \text{ and } x+h \\ f'(x)h &= f(x+h) - f(x) - \frac{f''(\xi)}{2}h^2 \\ f'(x) &= \frac{f(x+h)-f(x)}{h} - \frac{f''(\xi)}{2}h \end{aligned}$$

If we approximate

$$f'(x) \approx \frac{f(x+h)-f(x)}{h} \quad \text{the truncation error is } -\frac{f''(\xi)}{2}h$$

Errors in computing

Absolute and relative errors

Suppose y is exact result and $\widetilde{y}$ is an approximation for y

- Absolute error $|y - \widetilde{y}|$
- Relative error $|y - \widetilde{y}|/|y|$

Example 8. Suppose $y = 8.1472 \times 10^{-1}$ (accurate value), $\widetilde{y} = 8.1483 \times 10^{-1}$ (approximation). Then

$$|y - \widetilde{y}| = 1.1000 \times 10^{-4}, \qquad \frac{|y - \widetilde{y}|}{|y|} = 1.3502 \times 10^{-4}$$

Suppose $y = 1.012 \times 10^{18}$ (accurate value), $\widetilde{y} = 1.011 \times 10^{18}$ (approximation). Then

$$|y - \widetilde{y}| = 10^{15}, \qquad \frac{|y - \widetilde{y}|}{|y|} \approx 9.8814 \times 10^{-4} \approx 10^{-3}$$

Mean-value theorem

If $f \in C^1[a, b]$, $a < b$, then

$$f(b) = f(a) + (b - a)f'(\xi), \quad \text{for some } \xi \in (a, b)$$

From which

$$f'(\xi) = \frac{f(b) - f(a)}{b - a}$$

The Patriot disaster

During the Gulf War in 1992, a Patriot missile missed an Iraqi Skud, which killed 28 Americans. What happened?

- Patriot's internal clock counted tenths of a second and stored the result as an integer.
- To convert to a floating-point number, the time was multiplied by 0.1 stored in 24 bits.
- 0.1 in binary is $0.001\ 1001\ 1001\ \ldots$, which was chopped to 24 bits. Roundoff error $\approx 9.5 \times 10^{-8}$.
- After 100 hours the measured time had an error of

$$100 \times 60 \times 60 \times 10 \times 9.5 \times 10^{-8} \approx 0.34 \text{ seconds.}$$

- A Skud flies at $\approx 1,676$ meters per second. 0.34 seconds error results in

$$0.34 \times 1,676 \approx 569 \text{ meters.}$$