

Lecture 02: Sources of Error and basic error analysis

Outline

Sources of Error

Model/Data Error

Truncation Error

Floating Point Error

Basic definitions of Error Analysis

Truncation Error and Taylor's Theorem

Examples e^x , $1/x$

Propagation of Errors and Big O notation

Evaluation of Polynomials and Horner's method

Sources of Error

- 1) "Model Error": errors in model formulation or choice
e.g. Lotka-Volterra is a poor model for predator-prey interaction (fractional rabbits, no extinctions)
 - 1a) "Data Error": inaccuracies or uncertainties in model parameters

We won't really talk about Model error in this course but it is often the most serious source of error and difficult to control.

- 2) "Truncation Error": e.g. Errors arising from approximating a function with a simpler function e.g. $\sin(x) \approx x$
- 3) "Floating Point Error": Errors arising from approximating real numbers with finite-precision numbers

In general: All of these errors can contribute to the total error in an approximate numerical method (and can be hard to properly propagate through a "real problem").

Basic Definitions of Error Analysis

Given a true solution f and an approximate solution \hat{f} we define

Absolute error: $e = |f - \hat{f}|$

Relative error: $r = \frac{e}{|f|} = \frac{|f - \hat{f}|}{|f|}$

Decimal precision p (number of significant digits):

$$3 \times 10^{-3} < |r| < 5 \times 10^{-p}$$

Example:

$$f = \exp(1) = 2.718281828459046 \quad \hat{f} = 2.71$$

$$e = |f - \hat{f}| = 8.28 \times 10^{-3}$$

$$r \sim 3 \times 10^{-3}$$

$$p = 3$$

$$\hat{f} = 2.711345, \hat{\hat{f}} = 2.74$$

Truncation Error and Taylor's Theorem

Taylor's Theorem: Let $f(x)$ be a function $\in C^{n+1}[a, b]$ and $x_0 \in [a, b]$. Then for every $x \in (a, b)$ there exists a number $c = c(x)$ that lies between x_0 and x such that

$$f(x) = T_N(x) + R_N(x)$$

where

$$T_N(x) = \sum_{n=0}^N \frac{f^{(n)}(x_0)(x - x_0)^n}{n!}$$

$$f(x_0+h) = \sum \frac{f^{(n)} h^n}{n!}$$

and

$$R_N(x) = \frac{f^{(n+1)}(c)(x - x_0)^{n+1}}{(n+1)!}$$

$$\leq M (x-x_0)^{n+1} = M h^{n+1} = O(h^{n+1})$$

Example #1: $f(x) = e^x$ $x_0 = 0$

1) approximate $\hat{f}(x) = T_2(x)$

$$f = e^x \quad f' = e^x \quad f^{(n)} = e^x$$

$$T_n(x) = \sum_{n=0}^N \frac{x^n}{n!}$$

$$e^x \sim 1 + x + \frac{x^2}{2} + \varepsilon$$

$$R_N(x) = e^c \frac{x^{n+1}}{(n+1)!}$$

$x=1$

$$e^1 = 2.718 \dots$$

$$T_2(1) = 2.5$$

$$e \sim .2 \quad r \sim .1 \quad p = 1$$

$$R_n(x) \leq \frac{e}{6} \sim .5$$

Example #2: $f(x)=1/x$ $x_0=1$

1) approximate $\hat{f}(x)=T_2(x)$

$$f(x) = x^{-1} \quad f' = -x^{-2} \quad f'' = 2x^{-3} \quad f^{(n)} = \frac{(-1)^n n!}{x^{n+1}}$$

$$T_n(x) = \sum_{n=0}^N (-1)^n (x-1)^n$$

$$R_N(x) = \frac{(-1)^{N+1} (x-1)^{N+1}}{c^{N+1}} \quad h = x-1 \quad |R_N| < h^{N+1}$$

$$f(x) = 1/x$$

$$T_2(x) = 1 - h + h^2$$

$$+ R_N = -\frac{h^3}{c^3} \leq \frac{|h^3|}{c^3}$$

$$x=3 \quad x=1 \quad h=1$$

$$f(x) = 1/3 \quad f(x) = 1/1 \quad T_2(x) = .91 \quad r = 10^{-3} \quad p = 3$$

$$T_2(3) = 3$$

Rules of error propagation

in "Big O" notation let

$$f(h) = p(h) + O(h^n)$$

$$g(h) = q(h) + O(h^m)$$

and $r = \min(n, m)$ then:

$$f + g = p + q + O(h^r)$$

$$f \times g = pq + pO(h^m) + qO(h^n) + O(h^{n+m})$$

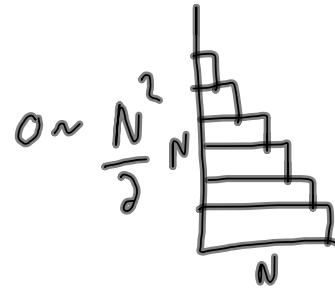
$$pq + O(h^r)$$

Horner's Method for Evaluation of Polynomials

$$\text{Given } P_N(x) = a_0 + a_1x + a_2x^2 + \dots + a_Nx^N$$

$$\text{or } = p_1x^N + p_2x^{N-1} + p_2x^{N-2} + \dots + p_{N+1}$$

what is the best way to evaluate P_N for any value of x ?



Consider $P_3(x)$ written two ways

$$P_3(x) = p_1x^3 + p_2x^2 + p_3x + p_4 \text{ and as "nested iteration"}$$

$$P_3(x) = (((p_1x + p_2)x + p_3)x + p_4) \quad 3 \sim O(N)$$

Operation counts:

Horner's Method for Evaluation of Polynomials Our first algorithm

Evaluate $y = P(x)$

given $p = [p_1 p_2 p_3 \dots p_{N+1}]$, and x

initialize:

$$y = p_1$$

for $n = 2, N+1$

$$y = y * x + p(n)$$

END

Horner's Method for Evaluation of Polynomials

Our first matlab function

```
function y = hornersPoly(p,x)
% hornersPoly - evaluates Polynomials using horner's rule
%   y = hornersPoly(p,x)
%
%   p:           - vector of polynomial coefficients such that
%                  $y(x) = P(1)x^n + P(2)x^{(n-1)} + \dots + P(n+1)$ 
%   x:           - values where polynomial is to be evaluated
%   y:           - outputs y(x)
```

```
global STUDENT_ID STUDENT_NAME;
STUDENT_ID = 'mws6';
STUDENT_NAME = 'Marc Spiegelman';
```

x = linspace(0,10)

```
y=p(1);
for i=2:length(p)
    y = y*x+p(i);
end
```

Horner's Method for Evaluation of Polynomials

Our second matlab function - use vectorized arguments

```
function y = hornersPolyVec(p,x)
% hornersPolyVec - evaluates Polynomials using horner's rule (vectorized arguments)
%   y = hornersPoly(p,x)
%
%   p:           - vector of polynomial coefficients such that
%                  $y(x) = P(1)x^n + P(2)x^{(n-1)} + \dots + P(n+1)$ 
%   x:           - vector of values where polynomial is to be evaluated
%   y:           - vector of outputs y(x)
```

```
global STUDENT_ID STUDENT_NAME;
STUDENT_ID = 'mws6';
STUDENT_NAME = 'Marc Spiegelman';
```

```
y=zeros(size(x));
y(:)=p(1);
for i=2:length(p)
    y = y.*x+p(i);
end
```

*help polyval |
y = polyval(P,x)*

This function is algorithmically equivalent to Matlab's polyval function (help polyval)

Limits of Truncation error: Floating point error