

CS321-002

**Introduction to Numerical
Methods**

Lecture 8

Final Review

Professor Jun Zhang

Department of Computer Science
University of Kentucky
Lexington, KY 40506-0046

December 7, 2000

Number Conversion

a general number should be converted in integer part and in fractional part separately

convert an integer number from base $\beta < 10$ to based 10

$$\begin{aligned}(21467)_8 &= 7 + 6 \times 8 + 4 \times 8^2 + 1 \times 8^3 + 2 \times 8^4 \\ &= 7 + 8(6 + 8(4 + 8(1 + 8(2)))) \\ &= 9015\end{aligned}$$

convert a fractional number from base $\beta < 10$ to based 10

$$\begin{aligned}(0.763)_8 &= 7 \times 8^{-1} + 6 \times 8^{-2} + 3 \times 8^{-3} \\ &= 8^{-3}(3 + 8(6 + 8(7))) \\ &= \frac{499}{512} \\ &= 0.9746\end{aligned}$$

Repeated Division

to convert a base 10 integer number to a base $\beta < 10$ number, we can use repeated division algorithm

$$\begin{array}{rcll} 2576/8 & = & 322 & \text{remainder } 0 \longrightarrow 0 \\ 322/8 & = & 40 & \text{remainder } 2 \longrightarrow 2 \\ 40/8 & = & 5 & \text{remainder } 0 \longrightarrow 0 \\ 5/8 & = & 0 & \text{remainder } 5 \longrightarrow 5 \end{array}$$

so we have $(2576.)_{10} = (5020.)_8$

it can be further converted to a binary number using the binary octal table

$$(2576.)_{10} = (5020.)_8 = (101\ 000\ 010\ 000.)_2$$

note that the first digit you obtain is the first digit next to the radix point

Integer-Fraction-Split

the integer-fraction-split process is used to convert a fractional base 10 number to a base β number

- 1.) multiply the (fractional) number by β ;
- 2.) take the integer part as the first (next) digit;
- 3.) repeat the process on the remaining *fractional part*.

$$0.372 \times 2 = 0.744 \longrightarrow 0$$

$$0.744 \times 2 = 1.488 \longrightarrow 1$$

$$0.488 \times 2 = 0.976 \longrightarrow 0$$

$$0.976 \times 2 = 1.952 \longrightarrow 1$$

$$0.952 \times 2 = 1.904 \longrightarrow 1$$

$$0.904 \times 2 = 1.808 \longrightarrow 1$$

hence, we have $(0.372)_{10} = (0.010111\dots)_2$

Normalized Scientific Notation

we write a number in the normalized scientific notation form as

$$728.359 = 0.728359 \times 10^3$$

only *machine numbers* can be represented exactly in a computer

to store a floating-point number using **32** bits

$$\pm q \times 2^m$$

sign of q needs 1 bit

integer $|m|$ needs 8 bits

number q needs 23 bits

single precision IEEE standard floating-point

$$(-1)^s \times 2^{c-127} \times (1.f)_2$$

Representation Errors

the relative error in correct rounding is

$$\frac{1}{2} \times 2^{-23}$$

the unit roundoff error is

$$\epsilon = 2^{-23}$$

floating-point representation

$$\mathbf{fl}(x) = x(1 + \delta) \quad \left(|\delta| \leq \frac{1}{2} \epsilon \right)$$

note that

$$\mathbf{fl}(x + y) = (x + y)(1 + \delta) \quad (|\delta| \leq 2^{-24})$$

avoid loss of significant digits (rationalization)

$$f(x) = \sqrt{x^2 + 1} - 1 \quad \text{at} \quad x \approx 0$$

$$f(x) = \frac{x^2}{\sqrt{x^2 + 1} + 1}$$

Taylor expansions for commonly used functions

Root Finding Algorithms

bisection method, Newton's method, and secant method

bisection method is slow, but is guaranteed to find a root

choose two initial points with different signs, compute the middle point and determine two new points based on the sign of the middle point

number of bisection steps for some tolerance

$$\frac{b - a}{2^{n+1}} < \epsilon$$

$$n > \frac{\log(b - a) - \log(2\epsilon)}{\log 2}$$

Newton's Method

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$$

Newton's method converges quadratically, while the bisection does linearly

Newton's method needs the first derivative value at each iteration

initial point must be sufficiently close to the solution point

Newton's method can be used to solve a system of linear equations and nonlinear equations

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - [\mathbf{f}'(\mathbf{x}^{(k)})]^{-1} \mathbf{f}(\mathbf{x}^{(k)})$$

where $\mathbf{f}'(\mathbf{x}^{(k)})$ is the Jacobian matrix

Secant Method

secant method tries to take the advantages of Newton's methods, but does not compute the derivative values

the derivative value is approximated by

$$f'(x_n) \approx \frac{f(x_{n-1}) - f(x_n)}{x_{n-1} - x_n}$$

the secant method generates iterates

$$x_{n+1} = x_n - \left(\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \right) f(x_n)$$

convergence rate is between that of Newton's method and bisection method, only one functional evaluation is needed at each iteration

need two initial points to start with, may use bisection method to generate the first iterate

Interpolation

given a set of (accurate) data, find a polynomial function of degree not higher than n that represents these data and possibly the missing data

$$\begin{array}{c|c|c|c|c} \mathbf{x} & \mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_n \\ \hline \mathbf{y} & \mathbf{y}_0 & \mathbf{y}_1 & \cdots & \mathbf{y}_n \end{array}$$

for Lagrange form, find the *cardinal functions*

$$l_i(x_j) = \delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

the Lagrange polynomial is

$$p_n(x) = \sum_{i=0}^n l_i(x) f(x_i)$$

the cardinal function is constructed as

$$l_i(x) = \prod_{j \neq i, j=0}^n \left(\frac{x - x_j}{x_i - x_j} \right) \quad (0 \leq i \leq n)$$

Newton Form

construct the polynomial step by step

$$p_{k+1}(x) = p_k(x) + c(x - x_0)(x - x_1) \cdots (x - x_k)$$

with

$$c = \frac{y_{k+1} - p_k(x_{k+1})}{(x_{k+1} - x_0)(x_{k+1} - x_1) \cdots (x_{k+1} - x_k)}$$

note that interpolation polynomials constructed by Lagrange or Newton form are the same. In fact, there is only one unique polynomial of degree not higher than n to interpolate a data set of $n + 1$

there are many higher order polynomials may satisfy the interpolation conditions, too

$$p(x_i) = y_i \quad \text{for} \quad 0 \leq i \leq n$$

Divided Difference

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

x	$f[]$	$f[,]$	$f[, ,]$	$f[, , ,]$
x_0	$f[x_0]$	$f[x_0, x_1]$	$f[x_0, x_1, x_2]$	$f[x_0, x_1, x_2, x_3]$
x_1	$f[x_1]$	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$	
x_2	$f[x_2]$	$f[x_2, x_3]$		
x_3	$f[x_3]$			

$$\begin{aligned}
 p_n(x) &= f[x_0] + f[x_0, x_1](x - x_0) \\
 &+ f[x_0, x_1, x_2](x - x_0)(x - x_1) \\
 &+ f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2) \\
 &+ \dots \\
 &+ f[x_0, \dots, x_n](x - x_0)(x - x_1) \cdots (x - x_{n-1})
 \end{aligned}$$

Derivative Approximation

sided $O(h)$ approximation of first derivative

$$f'(x) \approx \frac{1}{h}[f(x+h) - f(x)]$$

higher order approximations (Taylor expansions)

$$\begin{aligned} f(x+h) = & f(x) + h f'(x) + \frac{1}{2!} h^2 f''(x) \\ & + \frac{1}{3!} h^3 f'''(x) + \frac{1}{4!} h^4 f^{(4)}(x) + \dots \end{aligned}$$

$$\begin{aligned} f(x-h) = & f(x) - h f'(x) + \frac{1}{2!} h^2 f''(x) \\ & - \frac{1}{3!} h^3 f'''(x) + \frac{1}{4!} h^4 f^{(4)}(x) - \dots \end{aligned}$$

subtracting these two equations, we have

$$\begin{aligned} f(x+h) - f(x-h) = & 2h f'(x) \\ & + \frac{2}{3!} h^3 f'''(x) + \frac{2}{5!} h^5 f^{(5)}(x) + \dots \end{aligned}$$

2nd Order Approximation

$$f'(x) = \frac{1}{2h}[f(x+h) - f(x-h)] - \frac{h^2}{3!}f'''(x) - \frac{h^4}{5!}f^{(5)}(x) - \dots$$

after dropping the higher order terms, we have a second order approximation formula as

$$f'(x) \approx \frac{1}{2h}[f(x+h) - f(x-h)]$$

the leading truncated terms of this approximation scheme is $-\frac{h^2}{6}f'''(x)$. Hence the approximation is of $O(h^2)$. The approximation error goes to 0 as fast as $h^2 \rightarrow 0$. The exact truncation error is

$$-\frac{1}{6}h^2 \left[\frac{f'''(\xi_1) + f'''(\xi_2)}{2} \right] = -\frac{1}{6}h^2 f'''(\xi)$$

Richardson Extrapolation

Richardson extrapolation estimates the value of $\psi(0)$ from some computed values of $\psi(h)$ near 0

$$\psi(h) = f'(x) - a_2 h^2 - a_4 h^4 - a_6 h^6 - \dots$$

$$\psi\left(\frac{h}{2}\right) = f'(x) - a_2 \left(\frac{h}{2}\right)^2 - a_4 \left(\frac{h}{2}\right)^4 - a_6 \left(\frac{h}{2}\right)^6 - \dots$$

multiply the 2nd equation by 4 and subtract it from the 1st equation

$$\psi(h) - 4\psi\left(\frac{h}{2}\right) = -3f'(x) - \frac{3}{4}a_4 h^4 - \frac{15}{16}a_6 h^6 - \dots$$

hence

$$\psi\left(\frac{h}{2}\right) + \frac{1}{3} \left[\psi\left(\frac{h}{2}\right) - \psi(h) \right] = f'(x) + \frac{a_4}{4} h^4 + \frac{5}{16} a_6 h^6 + \dots$$

$f'(x)$ can be computed as accurate as $O(h^4)$

Interpolation Polynomial

we can approximate the function $f(x)$ by a polynomial $p_n(x)$ of order n , such that $p_n(x) \approx f(x)$

to compute $f'(x)$, we use the approximation $f'(x) \approx p'_n(x)$

higher order polynomials are avoided because of oscillation

let p interpolates f at two points, x_0 and x_1

$$p_1(x) = f(x_0) + f[x_0, x_1](x - x_0)$$

the first derivative of $p_1(x)$ is

$$p'_1(x) = f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \approx f'(x)$$

Trapezoid Rule

the interval $[a, b]$ is first divided into subintervals $[x_i, x_{i+1}]$, $0 \leq i \leq n - 1$. The basic trapezoid rule for the subinterval $[x_i, x_{i+1}]$ is

$$\int_{x_i}^{x_{i+1}} f(x) dx \approx \frac{1}{2}[f(x_i) + f(x_{i+1})](x_{i+1} - x_i)$$

the total area under the curve is

$$\begin{aligned} \int_a^b f(x) dx &\approx T(f; P) = \sum_{i=0}^{n-1} A_i \\ &= \frac{1}{2} \sum_{i=0}^{n-1} (x_{i+1} - x_i)[f(x_i) + f(x_{i+1})] \end{aligned}$$

for uniform spacing $h = (b - a)/n$, the composite trapezoid rule is

$$\begin{aligned} T(f; P) &= \frac{h}{2} \sum_{i=0}^{n-1} [f(x_i) + f(x_{i+1})] \\ &= h \left\{ \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}[f(x_0) + f(x_n)] \right\} \end{aligned}$$

Simpson's Rule

Simpson's rule is a three point numerical integration rule using the middle point of the interval with different weights for each point

$$\int_a^{a+2h} f(x) dx \approx \frac{h}{3} [f(a) + 4f(a+h) + f(a+2h)]$$

the error term of this approximation is

$$-\frac{h^5}{90} f^{(4)}(\xi)$$

for some point ξ in $(a, a + 2h)$. This should be compared to the error term of the simple trapezoid rule $O(h^3)$

Simpson's method can be used adaptively to refine the spacing in the subinterval that yields large error

strategy and criterion for adaptation: using Simpson's method on the whole interval and then on the two half-intervals

Gaussian Elimination

how to perform Gaussian elimination on a given linear systems, why the naive Gaussian elimination is not robust

what can we do to make Gaussian elimination more robust to solve an arbitrary linear system

what is the order of the long operation count in Gaussian elimination $n^3/3$

what is a tridiagonal system, what condition is sufficient to guarantee that there is no pivoting needed for solving a tridiagonal system

what is the advantages and disadvantages of iterative methods versus Gaussian elimination (direct method), what are the iterative method we studied, what type of linear systems that iterative methods may be advantageous

Spline Approximations

why we need spline approximations

what are the conditions for various degrees of spline, need to be able to verify these conditions

how can we say that a spline approximation is better than a polynomial interpolation

what is natural spline, how many arbitrary conditions are specified in a general cubic spline in order to obtain a natural spline $S''(t_0) = S''(t_n) = 0$

given a set of polynomials defined on separate intervals, figure out certain constants to make up a cubic or given spline

Least Squares

data are inaccurate, polynomial interpolation does not make sense

try to minimize the total error in a least squares sense

take the partial derivatives and set them to zero to obtain normal equations. Gaussian elimination may be used to solve the normal equations

least squares curve fitting may not pass through any of the given data points, but it is supposed to represent the true trend of the data

how to choose basis functions to make computation easier

what are the special polynomials we studied that can be used as a basis function for good curve fitting