OLD DOMINION UNIVERSITY

Numerical Differentiation

A. Godunov

1. Basics
2. Unequally spaced data
3. Equally spaced data
4. Taylor series approach
5. Error estimation

1

Part 1:

Basics of numerical differentiation

2

Analytic vs. numerical differentiation

Derivative

$$\frac{d}{dx}f(x) \equiv f'(x) \equiv f_x(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

In general, known functions can be differentiated exactly.

In numerical calculations the function $f(x)$ may be a known function, but normally $f(x)$ is given as a set of data points (a table of $[x, f(x)]$ pairs).

Differentiation of discrete data, requires an approximate numerical procedure.

3

3

Quick note

We cannot use

$$\frac{d}{dx}f(x) \equiv f'(x) \equiv f_x(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

for numerical differentiation because of the error in subtractive cancellation (a difference between two close numbers)

Besides, the subtractive cancellation is magnified by division on small number Δx.

And if Δx is not small enough then, for example, for $f(x) = a + bx^2$

$$f(x)' = \frac{f(x+\Delta x) - f(x)}{\Delta x} = 2bx + b\Delta x$$

We can only get the correct derivative $f'(x) = 2bx$ if $\Delta x \ll 2x$ otherwise the error is $b\Delta x$.

4

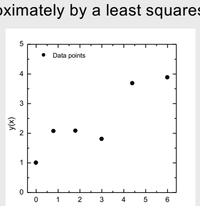
4

Key ideas

Most common numerical procedures for differentiation are based on

1. fitting an approximating function to the discrete data, or a subset of the discrete data
2. and the approximating polynomial is differentiated

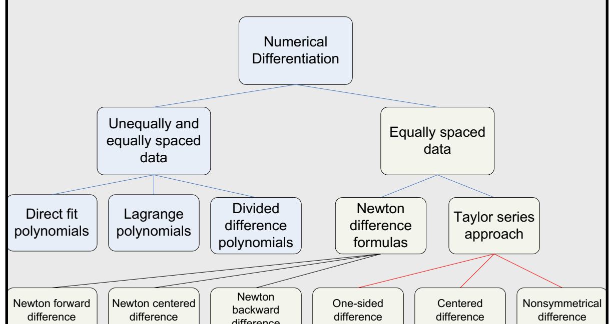
The polynomial may be fit exactly to a set of discrete data (see interpolation), or approximately by a least squares fit (see fitting data).



5

5

Overview



6

6

Part 2:

Unequally spaced data

7

Most common techniques

Three straightforward numerical differentiation procedures that can be used for both unequally spaced data and equally spaced data

The procedures are based on differentiation of fitting approximating functions

- Direct fit polynomials
- Lagrange polynomials
- Divided difference polynomials

8

8

Direct-fit polynomials

A direct fit polynomial procedure is based on fitting the data directly by a polynomial and differentiating the polynomial

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^n$$

where $P_n(x)$ is determined by one of the following methods:

1. Interpolation (if number of points $N = n + 1$)
2. The least-square fit (if $N > n + 1$)

After the approximating polynomial has been fit, the derivatives are determined by differentiating the approximating polynomial.

$$f'(x) \cong P_n'(x) = a_1 + 2a_2 x + 3a_3 x^2 + \cdots$$

$$f''(x) \cong P_n''(x) = 2a_2 + 6a_3 x + 12a_4 x^2 + \cdots$$

9

9

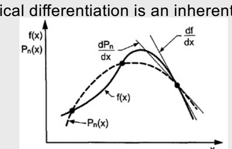
Numerical differentiation can be highly inaccurate!

Numerical differentiation formulas can be developed by fitting approximating functions (e.g., polynomials) to a set of discrete data and differentiating the approximating function. Thus,

$$\frac{d}{dx} f(x) \cong \frac{d}{dx} P_n(x)$$

However, even though the approximating polynomial $P_n(x)$ passes through the discrete data points exactly, the derivative of the polynomial may not be a very accurate approximation.

In general, numerical differentiation is an inherently inaccurate process.



10

10

Lagrange polynomials

The second procedure that can be used for both unequally spaced data and equally spaced data is based on differentiating a Lagrange polynomial. For example, consider the second- degree Lagrange polynomial,

$$P_2(x) = \frac{(x-b)(x-c)}{(a-b)(a-c)} f(a) + \frac{(x-a)(x-c)}{(b-a)(b-c)} f(b) + \frac{(x-a)(x-b)}{(c-a)(c-b)} f(c)$$

Differentiating yields:

$$f'(x) \cong P_2'(x) = \frac{2x-(b+c)}{(a-b)(a-c)} f(a) + \frac{2x-(a+c)}{(b-a)(b-c)} f(b) + \frac{2x-(a+b)}{(c-a)(c-b)} f(c)$$

$$f''(x) \cong P_2''(x) = \frac{2f(a)}{(a-b)(a-c)} + \frac{2f(b)}{(b-a)(b-c)} + \frac{2f(c)}{(c-a)(c-b)}$$

11

11

Divided Difference Polynomials

The third procedure that can be used for both unequally spaced data and equally spaced data is based on differentiating a divided difference polynomial (see interpolation),

$$P_n(x) = f_i^{(0)} + (x-x_0) f_i^{(1)} + (x-x_0)(x-x_1) f_i^{(2)} + (x-x_0)(x-x_1)(x-x_2) f_i^{(3)} + \cdots$$

Differentiating yields:

$$f'(x) \cong P_n'(x) = f_i^{(1)} + [2x-(x_0+x_1)] f_i^{(2)} + [3x^2 - 2(x_0+x_1+x_2)x + (x_0x_1 + x_0x_2 + x_1x_2)] f_i^{(3)} + \cdots$$

$$f''(x) \cong P_n''(x) = 2f_i^{(2)} + [6x - 2(x_0+x_1+x_2)] f_i^{(3)} + \cdots$$

12

12

Part 3:
Equally spaced data

13

Easier to work with equally spaced data

When the tabular data to be differentiated are known at equally spaced points, the Newton forward-difference and backward-difference polynomials, can be fit to the discrete data with much less effort than a direct fit polynomial, a Lagrange polynomial, or a divided difference polynomial.

This can significantly decrease the amount of effort required to evaluate derivatives.

14

Newton Forward-Difference Polynomial

Recall the Newton forward-difference polynomial,

$$P_n(x) = f_0 + s\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0 + \frac{s(s-1)(s-2)}{3!}\Delta^3 f_0 + \cdots + \text{Error}$$

$$\text{Error} = \binom{s}{h} h^{n+1} f^{n+1}(\xi) \quad x_0 \le \xi \le x_n$$

where the interpolating parameter is given by

$$s = \frac{x - x_0}{\Delta x} = \frac{x - x_0}{h}, \qquad x = x_0 + sh$$

Equation above requires that the approximating polynomial be an explicit function of x, whereas it is implicit in x. However,

$$f'(x) \cong \frac{d}{dx}P_n(x) = \frac{d}{ds}P_n(s)\,\frac{ds}{dx}, \qquad \frac{ds}{dx} = \frac{1}{h}, \qquad \text{then} \quad f'(x) \cong \frac{1}{h}\frac{d}{ds}P_n(s)$$

15

Newton Forward-Difference Polynomial

Substituting

$$P_n(x) = f_0 + s\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0 + \frac{s(s-1)(s-2)}{3!}\Delta^3 f_0 + \cdots$$

into

$$f'(x) \cong P_n'(x) = \frac{1}{h}\frac{d}{ds}P_n(s) \text{ gives}$$

$$P'_n(x) = \frac{1}{h}\left(\Delta f_0 + \frac{2s-1}{2}\Delta^2 f_0 + \frac{3s^2 - 6s + 2}{6}\Delta^3 f_0 + \cdots\right)$$

$$P_n''(x) = \frac{1}{h^2}(\Delta^2 f_0 + (s-1)\Delta^3 f_0 + \cdots)$$

Higher-order derivatives can be obtained in a similar manner. Recall that $\Delta^n f$ becomes less and less accurate as *n* increases. Consequently, higher-order derivatives become increasingly less accurate.

16

Newton Forward-Difference Polynomial

At $x = x_0, s = 0$ equations from the previous slide become

$$P'_n(x_0) = \frac{1}{h}\left(\Delta f_0 - \frac{1}{2}\Delta^2 f_0 + \frac{1}{3}\Delta^3 f_0 - \frac{1}{4}\Delta^4 f_0 \ldots\right)$$

$$P_n''(x) = \frac{1}{h^2}(\Delta^2 f_0 - \Delta^3 f_0 + \cdots)$$

Equations above are one-sided forward-difference formulas.

17

Errors

The error associated with numerical differentiation can be determined by differentiating the error term,

$$\frac{d}{dx}[\text{Error}(x_0)] = \frac{(-1)^n}{(n+1)}h^n f^{n+1}(\xi) \neq 0$$

Even though there is no error in $P_n(x_0)$, there is error in $P'_n(x_0)$.

The order of an approximation is the rate at which the error of the approximation approaches zero as the interval *h* approaches zero.

Analysis shows that

$$P_n'(x_0) \sim O(h^n), \qquad P_n''(x_0) \sim O(h^{n-1})$$

As we can see, each differentiation introduces an additional h into the denominator of the error term.

18

Newton centered-difference formulas

Centred-difference formulas can be obtained by evaluating the Newton forward-difference polynomial at points within the range of fit. For example, at $x = x_1, s = 1.0$

$$P_n'(x_1) = \frac{1}{h}\left(\Delta f_0 + \frac{1}{2}\Delta^2 f_0 - \frac{1}{6}\Delta^3 f_0 + \cdots\right)$$

$$P_n''(x) = \frac{1}{h^2}\left(\Delta^2 f_0 + \frac{1}{12}\Delta^4 f_0 + \cdots\right)$$

19

19

Difference formulas

The formulas for derivatives developed above are expressed in terms of differences.

Those formulas can be expressed directly in terms of function values if the order of the approximation is specified and the expressions for the differences in terms of function values are substituted into the formulas.

The resulting formulas are called *difference formulas.*

20

20

Difference formulas

Example for the one-sided forward-difference formula

$$P'_n(x_0) = \frac{1}{h}\left(\Delta f_0 - \frac{1}{2}\Delta^2 f_0 + \frac{1}{3}\Delta^3 f_0 - \frac{1}{4}\Delta^4 f_0 \ldots\right)$$

Keeping only the first term gives

$$P'_n(x_0) = \frac{1}{h}(\Delta f_0 + O(h^2))$$

Recall that $\Delta f_0 = (f_1 - f_0)$ then

$$P'_n(x_0) = \frac{f_1 - f_0}{h} + O(h)$$

Truncating after the second term and using $\Delta^2 f_0 = (f_2 - 2f_1 + f_0)$ gives

$$P_2'(x_0) = \frac{-3f_0 + 4f_1 - f_2}{2h} + O(h^2)$$

Higher-order difference formulas can be obtained in a similar manner. 21

21

Difference formulas (cont.)

Second order: one-sided forward-difference formula

$$P_2''(x_0) = \frac{f_0 - 2f_1 + f_2}{h^2} + O(h)$$

$$P_3''(x_0) = \frac{2f_0 - 5f_1 + 4f_2 - f_3}{h^2} + O(h^2)$$

Centered-difference formulas

$$P_2'(x_1) = \frac{f_2 - f_0}{2h} + O(h^2)$$

$$P_2''(x_1) = \frac{f_0 - 2f_1 + f_2}{h^2} + O(h^2)$$

Difference formulas of any order can be developed in a similar manner for derivatives of any order, based on one-sided, centered, or nonsymmetrical differences.

22

22

Part 4:

Taylor series approach

23

Taylor series and difference formulas

Difference formulas can also be developed using Taylor series.

This approach is especially useful for deriving finite difference approximations of exact derivatives (both total derivatives and partial derivatives) that appear in differential equations.

24

24

However …

Taylor series for a function f at point $x+h$ using x as a base point

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2!}f''(x)h^2 + \frac{1}{3!}f'''(x)h^3 + \cdots$$

then we can write for the first derivative at point x

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(x) - \frac{h^2}{6}f'''(x) + \cdots$$

If we keep the first term, then the error is proportional to h unless $f''(x) = 0$

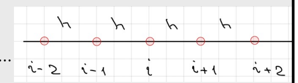
But, by making h smaller we can lose precision.

25

25

Taylor series

For, example, using points $i,\ i \pm 1$



$$f_{i+1} = f_i + f_i'h + \frac{1}{2}f_i''h^2 + \frac{1}{6}f_i'''h^3 + \cdots$$

$$f_{i-1} = f_i - f_i'h + \frac{1}{2}f_i''h^2 - \frac{1}{6}f_i'''h^3 + \cdots$$

difference $f_{i+1} - f_{i-1} = 2f_i'h + \frac{1}{3}f_i'''h^3 + \cdots$

First derivative

$$f_i' = \frac{f_{i+1} - f_{i-1}}{2h} - \frac{1}{6}f_i'''h^3$$

Now the error is $\sim h^2$, or one order better then before (check it for $f(x) = a + bx^2$)

sum $f_{i+1} + f_{i-1} = 2f_i + f_i''h^2 + \frac{1}{12}f_i''''h^4 + \cdots$ gives the second derivative

$$f_i'' = \frac{f_{i+1} + f_{i-1} - 2f_i}{h^2} - \frac{1}{12}f_i''''h^4$$

26

26

Partial derivatives

We can easily extend the Taylor series approach to partial derivatives for partial differential equations (PDE)

27

27

Taylor series – anything you want

1. Choose the order of derivatives n
2. Choose the order of the reminder in the difference h^m
3. Then the order of the reminder in Taylor series h^{m+n}
4. Choose the type: centered, forward, backward, nonsymmetric
5. Determine the number of additional grid points: $k \leq\ m+n-1$
6. Write Taylor series of order $m+n$ at the grid points
7. Combine the Taylor series and illuminate unwanted derivatives, then solve it.

Examples:

n=1, m=2	m+n-1=2	$i \pm 1$
n=1, m=4	m+n-1=4	$i \pm 1, i \pm 2$

Note: point i is always included in series.

28

28

Example 1

First derivative (n=1), fifth-order reminder h^5 (m=5),

then we need n+m-1=5: $i, i \pm 1, i \pm 2$

$$f_{i+1} = f_i + f_i'h + \frac{1}{2}f_i''h^2 + \frac{1}{6}f_i'''h^3 + \frac{1}{24}f_i''''h^4 + \frac{1}{120}f_i'''''h^5$$

$$f_{i-1} = f_i - f_i'h + \frac{1}{2}f_i''h^2 - \frac{1}{6}f_i'''h^3 + \frac{1}{24}f_i''''h^4 - \frac{1}{120}f_i'''''h^5$$

$$f_{i+2} = f_i + 2f_i'h + 2f_i''h^2 + \frac{4}{3}f_i'''h^3 + \frac{2}{3}f_i''''h^4 + \frac{4}{15}f_i'''''h^5$$

$$f_{i-2} = f_i - 2f_i'h + 2f_i''h^2 - \frac{4}{3}f_i'''h^3 + \frac{2}{3}f_i''''h^4 - \frac{4}{15}f_i'''''h^5$$

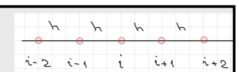
Then, multiple $(f_{i+1} - f_{i-1})$ by eight and and subtract $8(f_{i+1} - f_{i-1}) - (f_{i+2} - f_{i-2})$ will result in

$$f_i' = \frac{1}{12h}(f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2})$$

29

29

Example 2



Let's develop a third-order, nonsymmetrical, backward-biased, difference formula for $f'(x)$

First derivative (n=1), third-order reminder h^3 (m=3),

then n+m-1=3: $i \pm 1, i-2$

$$f_{i+1} = f_i + f_i'h + \frac{1}{2}f_i''h^2 + \frac{1}{6}f_i'''h^3 + \frac{1}{24}f_i''''h^4 + \cdots$$

$$f_{i-1} = f_i - f_i'h + \frac{1}{2}f_i''h^2 - \frac{1}{6}f_i'''h^3 + \frac{1}{24}f_i''''h^4 - \cdots$$

$$f_{i-2} = f_i - 2f_i'h + 2f_i''h^2 - \frac{4}{3}f_i'''h^3 + \frac{2}{3}f_i''''h^4 - \cdots$$

After some some manipulation with the series we will get

$$f_i' = \frac{1}{6h}(f_{i-2} - 6f_{i-1} + 3f_i + 2f_{i+1}) - \frac{1}{2}f''''(\zeta)h^3$$

30

30

Difference formulas (first order)

$$f_i' = \frac{f_{i+1} - f_i}{h} - \frac{1}{2}f''(\xi)h$$

$$f_i' = \frac{f_{i+1} - f_{i-1}}{2h} - \frac{1}{6}f'''(\xi)h^2$$

$$f'_i = \frac{-3f_i + 4f_{i+1} - f_{i+2}}{2h} + \frac{1}{3}f'''(\xi)h^2$$

The central difference gives better result when we use 2 points, or the same accuracy as forward/beachward difference with more points.

Therefore, it's often better to use the central difference, e.g. for h^4 we have

$$f_i' = \frac{f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2}}{12h} + \frac{1}{30}f'''''(\xi)h^4$$

31

31

Difference formulas (second order)

$$f''_i = \frac{f_i - 2f_{i+1} + f_{i+2}}{h^2} - f'''(\xi)h$$

$$f''_i = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} - \frac{1}{12}f''''(\xi)h^2$$

$$f''_i = \frac{2f_i - 5f_{i+1} + 4f_{i+2} - f_{i+3}}{h^2} + \frac{11}{12}f''''(\xi)h^2$$

The central difference gives better result when we use 3 points, or the same accuracy as forward/beachward difference with more points.

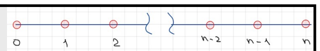
Again, it's often better to use the central difference,

$$f_i'' = \frac{-f_{i-2} + 16f_{i-1} - 30f_i + 16f_{i+1} - f_{i+2}}{12h^2} + \frac{1}{90}f''''''(\xi)h^4$$

32

32

Working with end-points



First-order derivatives

For the left end-point we use forward difference

$$f'_0 = \frac{-3f_0 + 4f_1 - f_2}{2h} - \frac{1}{3}f'''(\xi)h^2$$

$$f_0' = \frac{-11f_0 + 18f_1 - 9f_2 + 2f_3}{6h} - \frac{1}{4}f''''(\xi)h^3$$

and for the right end-point we use backward difference

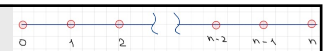
$$f_n' = \frac{f_{n-2} - 4f_{n-1} + 3f_n}{2h} + \frac{1}{3}f'''(\xi)h^2$$

$$f_n' = \frac{-2f_{n-3} + 9f_{n-2} - 18f_{n-1} + 11f_n}{6h} + \frac{1}{4}f''''(\xi)h^3$$

33

33

Working with end-points



Second-order derivatives

For the left end-point we use forward difference

$$f''_0 = \frac{f_0 - 2f_1 + f_2}{h^2} - f'''(\xi)h$$

$$f_0'' = \frac{2f_0 - 5f_1 + 4f_2 - f_3}{h^2} + \frac{11}{12}f''''(\xi)h^2$$

and for the right end-point we use backward difference

$$f''_n = \frac{f_{n-2} - 2f_{n-1} + f_n}{h^2} + f'''(\xi)h$$

$$f''_n = \frac{-f_{n-3} + 4f_{n-2} - 5f_{n-1} + 2f_n}{h^2} + \frac{11}{12}f''''(\xi)h^2$$

34

34

Example: for f(x)=sin(x) with h=0.2

x	f(x)=sin(x)	f'(x)=cos(x)	2 point Back	2 point C	4 point C
0	0.00000	1.00000			
0.2	0.19867	0.98007	0.99335	0.97355	
0.4	0.38942	0.92106	0.95375	0.91493	0.92101
0.6	0.56464	0.82534	0.87612	0.81984	0.82529
0.8	0.71736	0.69671	0.76357	0.69207	0.69667
1	0.84147	0.54030	0.62057	0.53671	0.54027
1.2	0.93204	0.36236	0.45284	0.35995	0.36234
1.4	0.98545	0.16997	0.26705	0.16884	0.16996
1.6	0.99957	-0.02920	0.07062	-0.02901	-0.02920
1.8	0.97385	-0.22720	-0.12863	-0.22569	-0.22719
2	0.90930	-0.41615	-0.32275	-0.41338	-0.41612
2.2	0.80850	-0.58850	-0.50401	-0.58459	-0.58847
2.4	0.67546	-0.73739	-0.66517	-0.73249	-0.73735
2.6	0.51550	-0.85689	-0.79981	-0.85119	-0.85684
2.8	0.33499	-0.94222	-0.90257	-0.93595	-0.94217
3	0.14112	-0.98999	-0.96934	-0.98341	-0.98994

35

35

Example: for f(x)=sin(x) with twice smaller h (h=0.1)

x	f(x)=sin(x)	f'(x)=cos(x)	2 point Back	2 point C	4 point C
0	0.00000	1.00000			
0.1	0.09983	0.99500	0.99833	0.99335	
0.2	0.19867	0.98007	0.98836	0.97843	0.98006
0.3	0.29552	0.95534	0.96851	0.95375	0.95533
0.4	0.38942	0.92106	0.93898	0.91953	0.92106
0.5	0.47943	0.87758	0.90007	0.87612	0.87758
0.6	0.56464	0.82534	0.85217	0.82396	0.82533
0.7	0.64422	0.76484	0.79575	0.76357	0.76484
0.8	0.71736	0.69671	0.73138	0.69555	0.69670
0.9	0.78333	0.62161	0.65971	0.62057	0.62161
1	0.84147	0.54030	0.58144	0.53940	0.54030
1.1	0.89121	0.45360	0.49736	0.45284	0.45359
1.2	0.93204	0.36236	0.40832	0.36175	0.36236
1.3	0.96356	0.26750	0.31519	0.26705	0.26750
1.4	0.98545	0.16997	0.21892	0.16968	0.16997
1.5	0.99749	0.07074	0.12045	0.07062	0.07074
1.6	0.99957	-0.02920	0.02079	-0.02915	-0.02920
1.7	0.99166	-0.12884	-0.07909	-0.12863	-0.12884
1.8	0.97385	-0.22720	-0.17817	-0.22682	-0.22720
1.9	0.94630	-0.32329	-0.27548	-0.32275	-0.32329
2	0.90930	-0.41615	-0.37003	-0.41545	-0.41615

36

36

Example: Fortran - first- or second-order derivatives

```
function deriv4(xx, xi, yi, ni, n, m)
!==================================================================
! Evaluate first- or second-order derivatives
! on three or four data points
! using interpolation based on divided differences
! written by: Alex Godunov
!------------------------------------------------------------------
! input ...
! xx    - the abscissa at which the derivative is to be evaluated
! xi()  - the arrays of data abscissas
! yi()  - the arrays of data ordinates
! ni - size of the arrays xi() and yi()
! n  - number of points to evaluate derivatives
! m  - order of a derivative (1 or 2)
! output ...
! deriv4  - the first- or second-order derivative
!=================================================================*/
implicit none
double precision deriv4, xx
integer ni, n, m
double precision xi(ni), yi(ni)
double precision x(n), f(n)
double precision d1, d2, d3, h, s
integer i, j, k, ix
```

37

Example (cont.)

```
! exit if too high-order derivative was needed,
! or too many base points were needed for derivatives
if (m > 2 .or. m>= n .or. n>5) then
  deriv4 = 0.0
  return
end if
! if x is ouside the xi(1)-xi(ni) interval set deriv4=0.0
if (xx < xi(1) .or. xx > xi(ni)) then
  deriv4 = 0.0
  return
end if
! a binary (bisectional) search to find i so that xi(i-1) < x <
xi(i)
i = 1
j = ni
do while (j > i+1)
  k = (i+j)/2
  if (xx < xi(k)) then
     j = k
    else
     i = k
  end if
end do
```

38

Example (cont.)

```
! shift i that will correspond to n-th order of interpolation
! the search point will be in the middle in x_i, x_i+1, x_i+2 ...
i = i + 1 - n/2
! check boundaries: if i is ouside of the range [1, ... n]-> shift i
if (i < 1) i=1
if (i + n > ni) i=ni-n+1

! just want to use index i
ix = i
! initialization of f(n) and x(n)
do i=1,n
  f(i) = yi(ix+i-1)
  x(i) = xi(ix+i-1)
end do
```

39

39

Example (cont.)

```
! calculate divided difference coefficients
d1 = f(2) - f(1)
if (n > 2) d2 = f(3) - 2.0*f(2) + f(1)
if (n > 3) d3 = f(4) - 3.0*f(3) + 3.0*f(2) - f(1)
h = x(2) - x(1)
s = (xx - x(1))/h
! calculate the first order derivative
if (m == 1) then
  deriv4 = (1.0/h)*d1
    if (n > 2) deriv4 = deriv4 + (1.0/h)*((2.0*s-1.0)/2.0)*d2
    if (n > 3) deriv4 = deriv4 + (1.0/h)*((3.0*s*s-6.0*s+2.0)/6.0)*d3
end if
! calculate the second order derivative
if (m == 2 .and. n > 2) then
  deriv4 = (1.0/h**2)*d2
  if (n > 3) deriv4 = deriv4 + (1.0/h**2)*(s-1.0)*d3
end if
end function deriv4
```

40

40

Example: Fortran – based on Lagrange interpolation

```
function deriv3(xx, xi, yi, ni, m)

!==================================================================
! Evaluate first- or second-order derivatives
! using three-point Lagrange interpolation
! written by: Alex Godunov
!------------------------------------------------------------------
! input ...
! xx    - the abscissa at which the interpolation is to be evaluated
! xi()  - the arrays of data abscissas
! yi()  - the arrays of data ordinates
! ni - size of the arrays xi() and yi()
! m  - order of a derivative (1 or 2)
! output ...
! deriv3  - interpolated value
!============================================================*/
implicit none
integer, parameter :: n=3
double precision deriv3, xx
integer ni, m
double precision xi(ni), yi(ni)
double precision x(n), f(n)
integer i, j, k, ix
```

41

Example (cont.)

```
! exit if too high-order derivative was needed,
if (m > 2) then
  deriv3 = 0.0
  return
end if
! if x is ouside the xi(1)-xi(ni) interval set deriv3=0.0
if (xx < xi(1) .or. xx > xi(ni)) then
  deriv3 = 0.0
  return
end if
! a binary (bisectional) search to find i so that xi(i-1)< x< xi(i)
i = 1
j = ni
do while (j > i+1)
  k = (i+j)/2
  if (xx < xi(k)) then
    j = k
  else
    i = k
  end if
end do
```

42

42

Example (cont.)

```
! shift i that will correspond to n-th order of interpolation
! the search point will be in the middle in x_i, x_i+1, x_i+2 ...
  i = i + 1 - n/2
! check boundaries: if i is ouside of the range [1, ... n]-> shift i
if (i < 1) i=1
if (i + n > ni) i=ni-n+1
ix = i
! initialization of f(n) and x(n)
do i=1,n
  f(i) = yi(ix+i-1)
  x(i) = xi(ix+i-1)
end do
! calculate the 1st order derivative using Lagrange interpolation
if (m == 1) then
 deriv3 =(2.0*xx - (x(2)+x(3)))*f(1)/((x(1)-x(2))*(x(1)-x(3)))
 deriv3 = deriv3 + (2.0*xx - (x(1)+x(3)))*f(2)/((x(2)-x(1))*(x(2)-x(3)))
 deriv3 = deriv3 + (2.0*xx - (x(1)+x(2)))*f(3)/((x(3)-x(1))*(x(3)-x(2)))
! calculate the 2nd order derivative using Lagrange interpolation
  else
    deriv3 =          2.0*f(1)/((x(1)-x(2))*(x(1)-x(3)))
    deriv3 = deriv3 + 2.0*f(2)/((x(2)-x(1))*(x(2)-x(3)))
    deriv3 = deriv3 + 2.0*f(3)/((x(3)-x(1))*(x(3)-x(2)))
end if
end function deriv3
```

43

Part 5:

Error estimation

44

Error estimation and extrapolation

The error can be estimated by evaluating the algorithm for two different step sizes h. The error estimate can be used both for error control and extrapolation (not only for derivatives).

Consider a numerical algorithm which approximates an exact calculation with an error that depends on an increment, h

$$f_{exact} = f(h) + Ah^n + Bh^{n+m} + Ch^{n+2m} + \cdots$$

where n is the order of the leading error term and m is the increment in the order of the following error terms. Applying the algorithm at two increment sizes, $h_1 = h$ and, gives $h_2 = h/R$ gives

$$f_{exact} = f(h) + Ah^n + O(h^{n+m})$$

$$f_{exact} = f(h/R) + A(h/R)^n + O(h^{n+m})$$

then, the difference

$$0 = f(h) - f(h/R) + Ah^n - A(h/R)^n + O(h^{n+m})$$

45

45

Error estimation and extrapolation (cont.)

$$0 = f(h) - f(h/R) + Ah^n - A(h/R)^n + O(h^{n+m})$$

$$0 = f(h) - f(h/R) + Ah^n(1 - 1/R^n) + O(h^{n+m})$$

Solving for the error term Ah^n gives

$$Error(h) = Ah^n = \frac{R^n}{R^n - 1}\left(f(h/R) - f(h)\right) + O(h^{n+m})$$

$$Error(h/R) = A(h/R)^n = \frac{1}{R^n - 1}\left(f(h/R) - f(h)\right) + O(h^{n+m})$$

The error estimates can be added to the approximate results to yield an improved approximation.

$$f_{exact} = f(h/R) + A(h/R)^n + O(h^{n+m})$$

This process is called *extrapolation*

$$Extrapolated\ value = f(h/R) + \frac{1}{R^n - 1}\left(f(h/R) - f(h)\right) + O(h^{n+m})$$

46

46