Review for Programming Exam and Final Exam

Larry Caretto
Mechanical Engineering 309

Numerical Analysis of Engineering Systems

May 5, 2014

California State University Northridge

Outline

- Programming and Final exams
  - VBA and MATLAB basics
  - Roots of equations
  - Matrix algebra and solution of simultaneous equations
  - Numerical differentiation
  - Interpolation
  - Regression
  - Quadrature
  - Numerical solution of ODEs

California State University Northridge

Programming Exam

- Can choose to use VBA or MATLAB
- Will have one relatively simple problem with two hours to get solution
  - Open book, notes, online help, but **no internet searches for code**
- Will have test cases with known solutions
  - Use test cases to verify program correctness
- Done with Excel workbook or commands from MATLAB command window

California State University Northridge 3

Programming Exam Rules

- Each student does own work and emails results to instructor
- No instructor help for programming
  - Can ask questions to clarify exam
  - Can get help for grave problems like computer crash
- Try to get as much done as possible
  - Describe future steps if you have not finished

California State University Northridge 4

Final Exam Reminder

- Monday, May 12, 8 to 10 pm, this room
- Closed book, no notes, no computer, no consultation, *etc*.
- Will be given necessary equations
  - If you think that some equation is missing ask and it will be provided
- Final will have same kinds of problems as midterm, with new algorithms mainly

California State University Northridge 5

Final Exam Problems

- Write simple VBA and MATLAB code (for general calculations or a given numerical algorithm)
- Given a numerical algorithm, evaluate a few steps with your calculator
- May be some short questions like how many data points does it take to fit a cubic polynomial or short exercises with matrices

California State University Northridge 6

Possible Numerical Algorithms

- Roots of equations, f(x) = 0, single equations only
- Simultaneous linear algebraic equations
- Interpolation
- Regression
- Numerical integration
- Numerical solution of Ordinary Differential Equations

California State University Northridge 7

Review VBA

- Option Explicit
- Dim and Const statements
- Expressions with operator precedence and replacement statements
  - Arithmetic, relational, logical and string operators
- Type conversion
  - Implicit as in MsgBox " x = " * x
  - Explicit with conversion functions like CDbl

California State University Northridge 8

Choice Statements

- The If statement

  If <condition> Then
  <statements to be executed if the condition is true>
  End If
  <Transfer control here if condition is false; normal transfer at end of if code>

  <condition> must have a Boolean value of true or false

- Alternative version for one statement in If

  If <condition> Then <statement>

California State University Northridge 9

If – Else If Explained

- If any condition is true, the statements following the If or Else If are executed
- Once those statements are executed controls to the first statement after the End If
- Statements for only the first true condition are executed
- The Else block is optional
  - If no conditions are true those statements are executed

If <condition1> The
<Statements don
Else If <condition2>
<Statements don
Else If <condition3>
<Statements don
<May be other cond
Else
<Statements don
End If
<Execute here after

California State University Northridge 10

Looping

- Count control loop repeats code a fixed number of time
- Conditional looping repeats while a condition is true or until a condition is false
- Both types of loops may be nested
- May use Exit For or Exit Do statements to exit loop before normal exit

California State University Northridge 11

Count Controlled Loop

For <counter> = <start> to <end>
<statements>
Next <counter>

If Step not specified, <increment> = 1

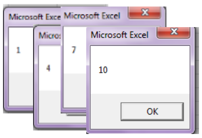
For <counter> = <start> to <end> _
Step <increment>
<statements>
Next <counter>

Statements in loop repeated nTimes = (<end> – <start>) /<increment> + 1

Loop not executed if nTimes <= 0

California State University Northridge 12

Count Controlled Examples




```
For k = 1 to 11 step 3
    msgBox k
Next k
```

```
x = 1 : term = x : sum = term
For n = 1 to 10
    term = term * x / n
    sum = sum + term
Next k
relErr = abs(sum/exp(x) - 1)
```

Code at left computes e^x for x = 1 with relative error of $1x10^{-8}$

California State University Northridge

13

Conditional Loop

<cond> is a condition (can be true or false)

<stmts> are statements executed in the loop (which should change the condition)

Do <stmts> if <cond> _ Then Exit Do <stmts> Loop	Do While <cond> <stmts> Loop	Do Until <cond> <stmts> Loop
	Do <stmts> Loop While <cond>	Do <stmts> Loop Until <cond>

Note tests before or after loop

California State University Northridge

14

Nested For Loops

- For loops used with arrays
- Nested for loops for 2D arrays

```
For k = n To 1 Step -1
  x(k) = a(n,n+1)
  For j = k+1 to n
    x(k) = x(k) - a(k,j) * x(j)
  Next j
Next k
```

k index in reverse order (from high to low)

What happens to the j loop, the first time in the k loop when k = n?

nTimes = [n - (n+1)]/1 +1 = 0, so loop is not executed

California State University Northridge

15

Arrays

- Arrays can be visualized as data on an experimental variable
  - Could describe pressure data points mathematically as P_1, P_2, *etc*.
  - In VBA we can represent these data points as P(1), P(2), *etc*.
  - We call the numbers (1, 2, *etc*.) indices or subscripts
    - We can use constants or variables for the subscripts: P(4), P(k), where k has a value

California State University Northridge

16

Two-dimensional Arrays

Consider an experiment where you vary the current over six levels, the voltage over four levels and measure the efficiency, e, of an electromechanical device. The data for each combination of current and voltage can be represented as shown below

	I(1)	I(2)	I(3)	I(4)	I(5)	I(6)
V(1)	e(1,1)	e(1,2)	e(1,3)	e(1,4)	e(1,5)	e(1,6)
V(2)	e(2,1)	e(2,2)	e(2,3)	e(2,4)	e(2,5)	e(2,6)
V(3)	e(3,1)	e(3,2)	e(3,3)	e(3,4)	e(3,5)	e(3,6)
V(4)	e(4,1)	e(4,2)	e(4,3)	e(4,4)	e(4,5)	e(4,6)

California State University Northridge

17

Dimensioning Arrays

- Can declare arrays as follows

```
Dim I(1 to 6) as double
Dim V(1 to 4) as double
Dim e(1 to 4, 1 to 6) as double
```

- Size below depends on Option Base

```
Dim I(6) as double
Dim V(4) as double
Dim e(4, 6) as double
```

What is lowest sub-script for these arrays?

Zero or one depending on Option Base

California State University Northridge

18

Using Arrays

- Arrays components are referenced by their subscripts
- This is often done in a For loop

```
For k = 0 to 100
      x(k) = sin(k * PI / 100)
Next k
```

- x is an array with 101 components giving sin(x) for $0 \le x \le \pi$, with $\Delta x = \pi/100$

California State University Northridge 19

Two-Dimensional Arrays

- Use nested for loops
  - Use example of current and voltages

```
For k = 1 to 4
   For j = 1 to 6
      Power(k,j) = I(j) * V(k)
   Next j
Next k
```

Recall table:
V was in rows
I was in columns
Power(k,j) is Power(row, column)
Are k and j indices correct?

California State University Northridge 20

Dynamic Arrays

- What if you do not know array size until program is actually running?
- Use Dim a() to tell compiler that a is an array then use ReDim with actual dimensions

```
Sub getArray( N as long) as Variant
Dim x() as Double : ReDim X(1 to N)
```

- Can go from Dim a() as Double to any size ReDim: ReDim a(1 to 10, 6 to 12)

California State University Northridge 21

Passing Arrays to Procedures

- Declare array in argument list with parentheses to indicate array

```
Sub mine( A() as double)
 'No dim statement for A
A(2,3) =
```

Use this for any size array. Variant arrays do not need ()

- Calling program sets actual dimensions on array and uses only the following

```
Dim B(1 to 10, 1 to 6) as double
Call mine(B)
```

California State University Northridge 22

Determining Array Bounds

- The UBound and LBound functions determine the upper and lower bounds of unknown array dimensions
- For a two-dimensional array, A(m,k)
  - LBound(A,1) is the lower bound of m
  - UBound(A,1) is the upper bound of m
  - LBound(A,2) is the lower bound of k
  - UBound(A,2) is the upper bound of k

California State University Northridge 23

Worksheet Arrays to VBA

- Passed as a range of cells
- First step is to set a type variant variable equal to the input range variable
  - The variant variable is now an two-dimensional array
  - May have single row or single column, but is still a two-dimensional array
  - Lower bound is always one
  - Can use UBound to get sizes

California State University Northridge 24

Worksheet Array Example

```
Function getMean (Ain As Range) _
                                    As Double
Dim A as Variant
Dim sum As double, cells As Long, k As Long
Dim nRows As Long, nCols As Long, m As Long
A = Ain   :   nRows = UBound(A,1) : sum = 0
nCols = UBound(A, 2)  : cells = nRows * nCols
For k = 1 To nRows
   For m = 1 To nCols
      sum = sum + A(k,m)
   Next m
```

Code from red line to end on next slide

Worksheet Array Example II

```
Dim sum As double, cells As Long, m As Long
Dim nRows As Long, nCols As Long, k As Long
A = Ain   :   nRows = UBound(A,1) : sum = 0
nCols = UBound(A, 2)  : cells = nRows * nCols
For k = 1 to nRows
   For m = 1 to nCols
      sum = sum + A(k,m)
   Next m
Next k
getMean = sum / cells
End Function
```


VBA Array to Worksheet

- VBA steps to return array to worksheet
  - Declare the function type as Variant
  - In the function or sub declare a working array for calculations
    - Use application.caller for dimensions
  - Write the code for values in working array
  - At end of function set <function name> = <working array name>
- To use the function: select cells; enter function in formula bar; Cont+Shift+Enter


```
Function array2wks(<arguments>) As Variant
  Dim userRows As Long
  Dim userColumns As Long
  Dim workArray() as Double
  'Statements below determine rows and columns
  userRows = Application.Caller.Rows.Count
  userColumns = Application.Caller.Columns.Count
  ReDim workArray(1 to userRows, 1 to userColumns)

    'Place code here to compute all
    'components of workArray

    array2wks = workArray
End Function
```


Strings

- Consider only variable length
- Use Dim str as String
- Use & or + as concatenation operator to join two strings
- Len(str) gives length of string
- Left, Right, and Mid give substrings in same manner as worksheet functions
- InStr function searches for substrings

Getting Programs to Work

- VBA detects syntax errors (one-line)
- Compilation (before execution) detects structure errors (more than one line)
- Programs will halt at many errors (like divide by zero)
- Programs will return errors like #NAME to worksheet instead of results
- Use test cases to make sure that a new program is working correctly

Getting Programs to Work

- Syntax errors: errors in single line
  - Line turns red after "completed" with optional error message and location
    - Select auto-syntax check
- Compilation errors: errors in program structure involving more than one line
  - *E.g.* If statement without following End If
  - *E.g.* Next statement without preceeding For
  - Could get "incorrect" error message for nested structures (see next slide)

California State University Northridge 31

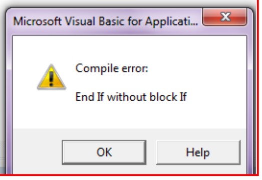
Misleading Error Message

- Error message when Next is omitted, but End If included to match If

```
Sub test()

    Dim k As Long
    If k > 10 Then
        For k = 6 To 5
    End If

End Sub
```



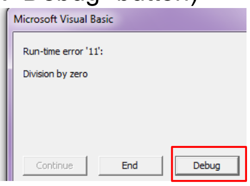
California State University Northridge 32

Identified Run-time Errors

- Syntax/compilation usually, but not always, easy to remedy
- Some run-time errors will stop and allow debugging (Click "Debug" button)

```
Sub test()

    Dim k As Long
    Dim r As Double
    Dim m As Long
    r = 1 / (k - m)
    If k > 10 Then
        m = 3
    End If
```



California State University Northridge 33

Run-time Error Highlighted

- After clicking "Debug" on previous dialog

```
Sub test()

    Dim k As Long
    Dim r As Double
    Dim m As Long
    r = 1 / (k - m)
    If k > 10 Then
        m = 3
    End If

End Sub
```

- Highlighted statement caused run-time error
  - Real error cause is values set for k and m are both zero
- Need to trace back from error statement to error cause

California State University Northridge 34

#NAME Error

- This error may be returned by a UDF when the function cannot be found
  - It was never defined
  - It is located in a different workbook
  - There is a module with the same name
  - The name is misspelled in the call
  - Arguments in the function call do not match arguments in the function header
  - Function is not located in a module (located on code for worksheet or ThisWorkbook)
  - Private Function located in a different module

California State University Northridge 35

#VALUE Error

- Returned to worksheet when there is an execution error that VBA cannot trap
  - Often linked to attempts to exceed array bounds
    - To find such errors use the debugger repeated times to find the statement causing the error
    - Locate area in code where execution halts for no apparent reason
    - Find exact statement where this error occurs
    - Hover mouse to find "out-of-range" arrays or other possible errors

California State University Northridge 36

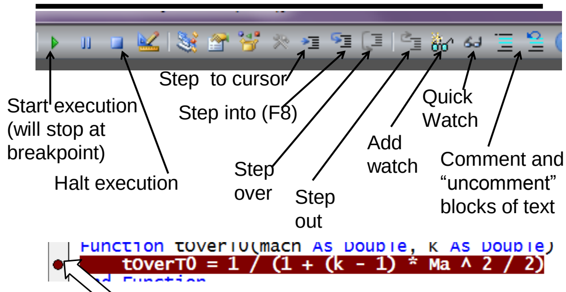
Incorrect Results

- Programs should always be tested with inputs whose solution is known
- If this solution is not found, use debugger to step through program to find errors
- Use worksheet to compute intermediate results to check against program values
  - Divided-difference table for polynomial interpolation as an example
    - Worksheet formulas may have errors too

Debugging

- Debugger allows you to step through a program and see intermediate values
  - Useful to find location of errors
- Items to use in debugger
  - Breakpoints stop execution at certain points
  - Step-by-step execution
  - Intermediate and Watch windows
  - Hover mouse over variable to get its value
  - Change statement to be executed next

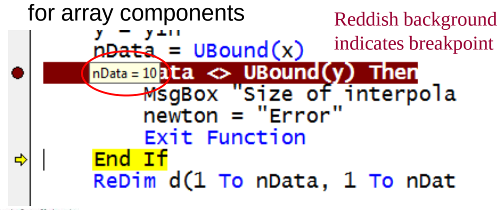
Useful Toolbar Icons



Right-click in left margin to set and clear breakpoints

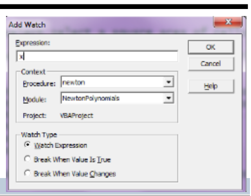
Hovering Mouse

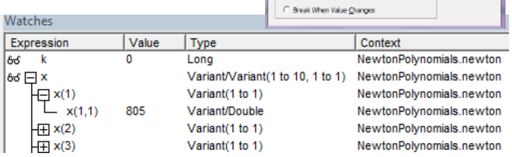
- Can hover mouse over scalars to show values of variables
  - Does not work for whole arrays, but works for array components



Watch Window

- Use "Add Watch" to specify variables and Watch Type
- View variables in Watch Window





Expression	Value	Type	Context
k	0	Long	NewtonPolynomials.newton
x		Variant/Variant(1 to 10, 1 to 1)	NewtonPolynomials.newton
x(1)		Variant(1 to 1)	NewtonPolynomials.newton
x(1,1)	805	Variant/Double	NewtonPolynomials.newton
x(2)		Variant(1 to 1)	NewtonPolynomials.newton
x(3)		Variant(1 to 1)	NewtonPolynomials.newton

Help

- Help systems for Excel and VBA
- Search function does not always return what you are looking for
- If you know the keyword, type it, place the cursor in the keyword, and press F1
- Sometimes a Google search for "VBA <subjectYouAreInterestedIn>" works

MATLAB Review I

- Ways of performing commands
  - Command window
    - Answers may be suppressed with semicolon
    - Default answer variable is ans
  - Functions and scripts
  - Anonymous functions
- Data entry commands for arrays
  - X = [1 2 3; 4 5 6; 7 8 9; 10 11 12]
    - Spaces between data on same row
    - Semicolons to start new row

MATLAB Review II

- Subarray commands x(r1:r2, c1:c2)
  - Use only : for complete row or column
- Other array definition: low:delta:high
  - Can omit delta if delta = 1
- Can get functions of arrays
  - t = 0:pi/100:2*pi; y = sin(t); plot(t,y)
- Transpose matrix: A^T in MATLAB is A'
  - Command A = [1 2 3]' gives $A = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

MATLAB Review III

- Matrix operations (+ - * / ^)
- Term-by-term operations (+ - .* ./ .^)
- Valid operations between matrix, **X**, and scalar a: a + **X**, a **-** **X**, a * **X**, a ./ **X**, **X**/a
- Can create larger matrices from smaller ones if they are compatible
  - C = [A B] if A and B have same rows
  - C = [A; B] if A and B have same columns

Trajectory Function Example

```
function [x, y] = traj(v0, theta, N)
%Computes frictionless trajectory
%Uses SI units (meters, seconds)
%V0 is initial speed in m/s
%theta is initial angle in degrees
%N is number of points computed
g = 9.80665;    %gravity in m/s^2
tMax = 2 * v0 * sind(theta) / g;
t = 0:tMax/(N-1):tMax;
x = v0*cosd(theta) * t;
y = v0*sind(theta) * t - g * t .^ 2/2;
plot(x,y);
end
```

Use file names the same as the function names (e.g. traj.m) to save functions

Array operation

Use semicolons to avoid intermediate output from function code

Using Your MATLAB Functions

- Used as any MATLAB Function

```
>>v0 = 10;
>>theta = 60;
>>N = 100;
>>[x, y] = traj(v0, theta, N);
```

- Can use only part of return variables
  - >>= traj(v0,theta,N) returns x values in ans
  - >>x = traj(V0,theta,N) returns x values in x

MATLAB if Statements

- Use the following format

```
if <expression1>
    <statements1>
elseif <expression2>
    <statements2>
<other elseif’s possible here>
else    %optional
    <statements>
end
```

Same structure as VBA but “Then” not used

All keywords (if, else, elseif) in lower case

Final statement is end, not endif

MATLAB While Statement

- MATLAB has only one conditional looping command with a test before

```
while <condition>
    <statements>
end
```

MATLAB keywords (while and end) must be lower case

- The `<statements>` in the while loop continue to execute while the `<condition>` is true

California State University Northridge 49

MATLAB for Statement

- Similar to VBA For statement, but loop limits are a MATLAB array specification

```
for <index> = <MATLAB array>
    <statements>
end
```

- Examples of for statements

```
for T = [300, 500, 1000, 5000]
for x = 0 : 0.01 : 2
for k = 1 : 25
```

(Same as 1:1:25)

California State University Northridge 50

Review Roots of Equations

- Write equation in form f(x) = 0
- Methods solving f(x) = 0
  - Bisection
  - Secant method
  - Newton's Method
  - False position (*regula falsi*)
  - Successive substitution
  - May be given algorithm for other method and asked to apply it

California State University Northridge 51

Methods and Process

- Bisection and False Position require two initial guesses that bracket root
- Newton's method requires one guess
- Secant method requires two guesses (do not have to bracket root)
- Different convergence conditions
  - Absolute error in Δx or f(x)
  - Relative error in Δx
  - Combination of above

California State University Northridge 52

Matrix Basics

- Define an m by n matrix as an array of with m rows and n columns

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$$

  - Row, column, diagonal, unit, null, inverse and transpose matrices
- Matrix equality, addition, subtraction require same size matrices

California State University Northridge 53

General Matrix Multiplication

- For matrix multiplication, **C** = **AB**
  - **A** has n rows and p columns
  - **B** has p rows and m columns
  - **C** has n rows and m columns

$$c_{ij} = \sum_{k=1}^{p} b_{ik} a_{kj}$$

$$(i = 1, n;\ j = 1, m)$$

- Example

$$\mathbf{A} = \begin{bmatrix} 3 & 0 & -6 \\ 4 & -2 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 3 & 4 \\ 1 & 2 \\ 6 & 1 \end{bmatrix}$$

$$\mathbf{AB} = \begin{bmatrix} 3(3)+0(1)-6(6) & 3(4)+0(2)-6(1) \\ 4(3)-2(1)+0(6) & 4(4)-2(2)+0(1) \end{bmatrix} = \begin{bmatrix} -27 & 6 \\ 10 & 12 \end{bmatrix}$$

California State University Northridge 54

From Equations to **Ax** = **b**

- Usual form for N = 3 equations

```
3x + 7y - 3z =  8
2x - 4y +  z = -3
8x + 6y - 2z = 14
```

$$\underbrace{\begin{bmatrix} 3 & 7 & -3 \\ 2 & -4 & 1 \\ 8 & 6 & -2 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 3x_1 & 7x_2 & -3x_3 \\ 2x_1 & -4x_2 & 1x_3 \\ 8x_1 & 6x_2 & -2x_3 \end{bmatrix}}_{\mathbf{Ax}} = \underbrace{\begin{bmatrix} 8 \\ -3 \\ 14 \end{bmatrix}}_{\mathbf{b}}$$

- An equation is a row in the **Ax** = **b** format

California State University Northridge 55

Gaussian Elimination

- Solve the set of equations on the right

$$2x_1 - 4x_2 - 26x_3 = -34 \ (i)$$
$$-3x_1 + 2x_2 + 9x_3 = 13 \ (ii)$$
$$7x_1 + 3x_2 + 8x_3 = 14 \ (iii)$$

- Subtract –3/2 times (i) from equation (ii) and 7/2 times (i) from (iii)

$$\left[-3 - \left(\frac{-3}{2}\right)2\right]x_1 + \left[2 - \left(\frac{-3}{2}\right)(-4)\right]x_2 + \left[9 - \left(\frac{-3}{2}\right)(-26)\right]x_3 = \left[13 - \left(\frac{-3}{2}\right)(-34)\right]$$

$$\left[7 - \left(\frac{7}{2}\right)2\right]x_1 + \left[3 - \left(\frac{7}{2}\right)(-4)\right]x_2 + \left[8 - \left(\frac{7}{2}\right)(-26)\right]x_3 = \left[14 - \left(\frac{7}{2}\right)(-34)\right]$$

Unnecessary computer operations

California State University Northridge 56

Gaussian Elimination II

- Result from first set of operations

$$2x_1 - 4x_2 - 26x_3 = -34$$
$$0x_1 - 4x_2 - 30x_3 = -38$$
$$0x_1 + 17x_2 + 99x_3 = 133$$

- Subtract 17/(-4) times (ii) from (iii)

$$x_2\left[17 - \left(\frac{17}{-4}\right)(-4)\right] + x_3\left[99 - \left(\frac{17}{-4}\right)(-30)\right] = \left[133 - \left(\frac{17}{-4}\right)(-38)\right]$$

- Final upper-triangular form

$$2x_1 - 4x_2 - 26x_3 = -34$$
$$0x_1 - 4x_2 - 30x_3 = -38$$
$$0x_1 + 0x_2 - \frac{57}{2}x_3 = -\frac{57}{2}$$

California State University Northridge 57

Back Substitution

- Final upper-triangular form

$$2x_1 - 4x_2 - 26x_3 = -34$$
$$-4x_2 - 30x_3 = -38$$
$$-\frac{57}{2}x_3 = -\frac{57}{2}$$

- Solve third equation for x_3

$$x_3 = -\frac{57}{2} \Big/ -\frac{57}{2} = 1$$

- Solve second equation for x_2

$$x_2 = \frac{-38 + 30x_3}{-4} = \frac{-38 + 30(1)}{-4} = 2$$

- Solve first equation for x_1

$$x_1 = \frac{-34 + 4x_2 + 26x_3}{2} = \frac{-34 + 4(2) + 26x_3(1)}{2} = 0$$

California State University Northridge 58

Solutions for **Ax** = **b**

- For a set of n equations in n unknowns
  - If Rank(**A**) = Rank([**A b**]) = n there is a unique solution 2x + y = 4; 2x – y = 0
  - If Rank(**A**) = Rank([**A b**]) < n: an infinite number of solutions x + y = 1; 2x + 2y = 2
  - If Rank(**A**) ≠ Rank([**A b**]) there are no solutions x + y = 1; 2x + 2y = 3
- Use Gaussian elimination to find Rank as number of nonzero rows

California State University Northridge 59

Numerical Differentiation

- Formulas have following properties
  - Type of derivative (first, second, third, *etc*.)
  - Direction of points used in the derivative, relative to the point of the derivative (forward, backward, central)
  - Order of the error: $O(h^n)$ is an n^{th} order error (truncation error proportional to h^n)
- Roundoff error occurs when h is so small that significant figures are lost

California State University Northridge 60

Some Derivative Expressions

$$f_i' = \frac{f_{i+1} - f_i}{h} + O(h) \qquad f_i' = \frac{f_i - f_{i-1}}{h} + O(h)$$

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

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

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

Note order of derivative, order of error, and direction (forward vs. backward)

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

More Derivative Expressions

$$f_i'' = \frac{2f_i - 5f_{i-1} + 4f_{i-2} - f_{i-3}}{h^2} + O(h^2)$$

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

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

- What is sum of coefficients in numerator for each expression?
- Is there a reason for this?

Interpolation

- Given a table of data, (x_i, y_i) estimate a value of y for an x value not in the table
- Use N+1 table (x_i, y_i) points for N^th^-order polynomial
- Pick points that surround the value of x for which the polynomial is to be evaluated
- Get Newton polynomial from divided difference table

Divided Difference Table

x_0	y_0	$\leftarrow a_0$		
		$F_0 = \frac{y_1 - y_0}{x_1 - x_0}$	$\leftarrow a_1$	
x_1	y_1		$S_0 = \frac{F_1 - F_0}{x_2 - x_0}$	$\leftarrow a_2$
		$F_1 = \frac{y_2 - y_1}{x_2 - x_1}$		$T_0 = \frac{S_1 - S_0}{x_3 - x_0}$
x_2	y_2		$S_1 = \frac{F_2 - F_1}{x_3 - x_1}$	$\uparrow a_3$
		$F_2 = \frac{y_3 - y_2}{x_3 - x_2}$		
x_3	y_3			

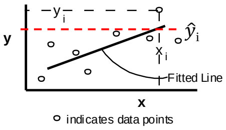
Divided Difference Example

0	0	$\leftarrow a_0$		
		$F_0 = \frac{10-0}{10-0} = 1$	$\leftarrow a_1$	
10	10		$S_0 = \frac{3-1}{20-0} = .1$	$\leftarrow a_2$
		$F_1 = \frac{40-10}{20-10} = 3$		$T_0 = \frac{.15-.1}{30-0} = \frac{1}{600}$
20	40		$S_1 = \frac{6-3}{30-10} = .15$	$\uparrow a_3$
		$F_2 = \frac{100-40}{30-20} = 6$		
30	100			

Divided Difference Example II

- Divided difference table gives $a_0 = 0$, $a_1 = 1$, $a_2 = .1$, and $a_3 = 1/600$
- Polynomial $p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2)$ $= 0 + 1(x - 0) + 0.1(x - 0)(x - 10) + (1/600)(x - 0)(x - 10)(x - 20) = x + 0.1x(x - 10) + (1/600)x(x - 10)(x - 20)$
- Check $p(30) = 30 + .1(30)(20) + (1/600)(30)(20)(10) = 30 + 60 + 10 = 100$ (Correct!)

Linear Regression



- Seeks approximate linear relationship among data set (x_i, y_i)
- Fit equation: $\hat{y}_i = a + bx_i$
- Notation $\hat{y}_i$ indicates approximate value, which may be different from data y_i
- Equations for a and b based on minimizing sum of squares of differences between actual and approximate data

$\sum(y_i - \hat{y}_i)^2 \rightarrow$ Minimum

California State University Northridge 67

Equations for a and b

- Substitute equation for a into equation for b (both copied below) and solve for b

$$a = \frac{\sum_{i=1}^{N} y_i - b\sum_{i=1}^{N} x_i}{N}$$

$$2\sum_{i=1}^{N} x_i y_i - 2a\sum_{i=1}^{N} x_i - 2b\sum_{i=1}^{N} x_i^2 = 0$$

$$b = \frac{N\sum_{i=1}^{N} x_i y_i - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} y_i\right)}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2} = \frac{\sum_{i=1}^{N} x_i y_i - N(\bar{x})\bar{y}}{\sum_{i=1}^{N} x_i^2 - N(\bar{x})^2}$$

- First solve for b then solve for a
  - Can set a = 0 to force line through origin
- Can use equations with all sums or means

California State University Northridge 68

Confidence Limits

- R^2 value gives overall measure of fit
  - $0 \le R^2 \le 1$
  - Confidence limits for the regression parameters a and b based on t statistic and user-specified confidence limit $1 - \alpha$
    - Typically choose $\alpha = .05$ for 95% confidence

$$b \pm t_{\alpha/2,n-2}\frac{s_{y|x}}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x}_i)^2}} \qquad a \pm t_{\alpha/2,n-2}\, s_{y|x}\sqrt{\frac{1}{n} + \frac{(\bar{x})^2}{\sum_{i=1}^{N}(x_i - \bar{x})^2}}$$

Standard errors for a and b

California State University Northridge 69

Multivariate Linear Regression

- In general we can have K predictive variables, x_1 to x_k
- General model equation: $y = b_0 + \sum_{j=1}^{K} b_j x_j$
- How do we represent the data?
  - Each data set consists of one value of y and one value for each of the x_j variables
  - For data set m, we can call the value of y, y_m, and we can call the value of x_j for data set m x_{jm}
  - Multivariate analysis finds coefficients b_0 to b_K
  - Each coefficient has standard error (times t statistic = confidence interval)

California State University Northridge 70

More Equations to be Used

- Compute estimated y values $\hat{y}_m = b_0 + \sum_{j=1}^{K} b_j x_{jm}$
- Compute the R^2 value

$$R^2 = 1 - \frac{\sum_{m=0}^{N-1}(y_m - \hat{y}_m)^2}{\left(\sum_{m=0}^{N-1} y_m^2\right) - N\,(\bar{y})^2}$$

California State University Northridge 71

Numerical integration formulas

- Trapezoid Rule

$$I = \int_a^b f(x)dx = T + E = h\left[\frac{f_0 + f_N}{2} + \sum_{i=1}^{N-1} f_i\right] + O(h^2)$$

- Simpson's (1/3) rule (even N only)

$$I = \int_a^b f(x)dx = S + E = \frac{h}{3}\left[f_0 + f_N + 4\sum_{i=1,3,5}^{N-1} f_i + 2\sum_{i=2,4,6}^{N-2} f_i\right] + O(h^4)$$

- Basic definitions of step size, h, number of intervals, N, $x_0 = a$, $x_N = b$, $f_k = f(a + kh)$

$$h = \frac{b-a}{N} \qquad x_k = a + kh \qquad f_k = f(x_k)$$

California State University Northridge 72

Richardson Extrapolation

- Have numerical expression, F, for two different step sizes h and kh
  - Call these F(h) and F(kh)
  - These expressions have a lead error term with order n, $O(h^n)$ (error proportional to h^n)
  - Can get higher order expression by using Richardson Extrapolation formula
    - Typically pick k > 1, but it could be < 1 so long as formula is consistently applied

$$RE = \frac{k^n F(h) - F(kh)}{k^n - 1}$$

73

Romberg Integration

- General forms for initial T_{k0} and subsequent T_{km}

$$T_{km} = \frac{4^m T_{k,m-1} - T_{k-1,m-1}}{4^m - 1} \quad m = 1, \cdots, k; \text{ all } k$$

$$T_{k0} = T(h_{init}/2^k) \quad \text{all } k$$

$$h_{init} = \frac{b-a}{2}$$

$$T_{11} = \frac{4^1 T_{10} - T_{00}}{4^1 - 1}$$

$$T_{21} = \frac{4^1 T_{20} - T_{10}}{4^1 - 1} \quad T_{22} = \frac{4^2 T_{21} - T_{11}}{4^2 - 1}$$

$$T_{31} = \frac{4^1 T_{30} - T_{20}}{4^1 - 1} \quad T_{32} = \frac{4^2 T_{31} - T_{21}}{4^2 - 1} \quad T_{33} = \frac{4^3 T_{32} - T_{22}}{4^3 - 1}$$

T(h) = Trapezoid rule result

74

Numerical ODE Solution

- Solve initial value problem, dy/dx = f(x,y) (known) with $y(x_0) = y_0$
  - Use a finite difference grid: $x_{i+1} - x_i = h_{i+1}$
  - Replace derivative by finite-difference approximation: $dy/dx \approx (y_{i+1} - y_i) / (x_{i+1} - x_i) = (y_{i+1} - y_i) / h_{i+1}$
  - Derive a formula to compute f_{avg} the average value of f(x,y) between x_i and x_{i+1}
  - Replace dy/dx = f(x,y) by $(y_{i+1} - y_i) / h_{i+1} = f_{avg}$
  - Repeatedly compute $y_{i+1} = y_i + h_{i+1} f_{avg}$

75

Order of ODE Methods

- Local error is error after one step when the initial conditions are known exactly
- Global error is the error after more than one step
- For an n^{th}-order local error, the global error has an order of n – 1
- The global error is the more important error which is used to describe a method

76

Review Notation and Order

- x_i is independent variable
- y_i is numerical solution at $x = x_i$
- f_i is derivative found from x_i y_i: $f_i = f(x_i, y_i)$
- $y(x_i)$ is the exact value of y at $x = x_i$
- $f(x_i, y(x_i))$ is the exact derivative
- $e_1 = y(x_1) - y_1$ = local truncation error
- $E_j = y(x_j) - y_j$ is global truncation error
- If e is $O(h^n)$, then E is $O(h^{n-1})$

77

Review Simple Methods

- Euler: $y_{i+1} = y_i + h_i f_i = y_i + h_i f(x_i, y_i)$ — First order
- Huen's method (second order)

$$y_{i+1}^0 = y_i + h_{i+1} f(x_i, y_i) \qquad x_{i+1} = x_i + h_{i+1}$$

$$y_{i+1} = y_i + \frac{h_{i+1}}{2}\left[f(x_i, y_i) + f(x_{i+1}, y_{i+1}^0)\right] = \frac{y_i + y_{i+1}^0 + h_{i+1} f(x_{i+1}, y_{i+1}^0)}{2}$$

- Modified Euler method (second order)

$$y_{i+\frac{1}{2}} = y_i + \left[\frac{h_{i+1}}{2}\right] f(x_i, y_i) \qquad x_{i+\frac{1}{2}} = x_i + \frac{h_{i+1}}{2}$$

$$y_{i+1} = y_i + h_{i+1} f(x_{i+\frac{1}{2}}, y_{i+\frac{1}{2}})$$

78

Review 4th Order Runge-Kutta

- Uses four derivative evaluations per step

$$y_{i+1} = y_i + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \qquad x_{i+1} = x_i + h_{i+1}$$

$$k_1 = h_{i+1} f(x_i, y_i)$$

$$k_2 = h_{i+1} f\left(x_i + h_{i+1}/2, y_i + k_1/2 \right)$$

$$k_3 = h_{i+1} f\left(x_i + h_{i+1}/2, y_i + k_2/2 \right)$$

$$k_4 = h_{i+1} f(x_i + h_{i+1}, y_i + k_3)$$

California State University Northridge 79

Systems of ODEs

- Can convert n^{th} order ODE into n first-order ODEs
- Can apply algorithms for one first-order ODE to systems of first-order ODEs
  - Must have initial conditions on all variables
  - Converting an n^{th} order ODE to n first-order ODEs gives n – 1 derivative ODEs whose initial values we need
  - Must apply each step of algorithms to all ODEs before going on to next step

California State University Northridge 80

Example

- Two masses joined by a spring/damper
- Original ODEs for each mass

$$m_1 \frac{d^2x_1}{dt^2} + c\left(\frac{dx_1}{dt} - \frac{dx_2}{dt}\right) + k(x_1 - x_2) = F_1$$

$$m_2 \frac{d^2x_2}{dt^2} + c\left(\frac{dx_2}{dt} - \frac{dx_1}{dt}\right) + k(x_2 - x_1) = F_2$$

- Define velocities $\frac{dx_1}{dt} = v_1 \qquad \frac{dx_2}{dt} = v_2$
- Rewrite original ODEs using velocities

$$\frac{dv_1}{dt} + \frac{c}{m_1}(v_1 - v_2) + \frac{k}{m_1}(x_1 - x_2) = \frac{F_1}{m_1}$$

$$\frac{dv_2}{dt} + \frac{c}{m_2}(v_2 - v_1) + \frac{k}{m_2}(x_2 - x_1) = \frac{F_2}{m_2}$$

California State University Northridge 81

Example Continued

- Replace x_1, x_2, v_1, v_2 in equations below by y_1, y_2, y_3, y_4

$$\frac{dx_1}{dt} = v_1 \qquad \frac{dv_1}{dt} + \frac{c}{m_1}(v_1 - v_2) + \frac{k}{m_1}(x_1 - x_2) = \frac{F_1}{m_1}$$

$$\frac{dx_2}{dt} = v_2 \qquad \frac{dv_2}{dt} + \frac{c}{m_2}(v_2 - v_1) + \frac{k}{m_2}(x_2 - x_1) = \frac{F_2}{m_2}$$

- Result is standard-form system: $dy_k/dt = f_k$

$$\frac{dy_1}{dt} = f_1 = y_3 \qquad \frac{dy_3}{dt} = f_3 = \frac{F_1}{m_1} - \frac{c}{m_1}(y_3 - y_4) - \frac{k}{m_1}(y_1 - y_2)$$

$$\frac{dy_2}{dt} = f_2 = y_4 \qquad \frac{dy_4}{dt} = f_4 = \frac{F_2}{m_2} - \frac{c}{m_2}(y_4 - y_3) - \frac{k}{m_2}(y_2 - y_1)$$

California State University Northridge 82

ODE Systems

- Convert system of higher order equations into system of first order ODEs
  - Do this by defining new variables for all higher-order derivatives (order > 1)
    - These definitions become simple ODEs for the resulting system
  - Must have initial conditions on all derivatives so created
  - N ODE system form $dy_k/dt = f_k(t, y_1,\ldots y_N)$
    - Write function/sub to compute all f_k

California State University Northridge 83

ODE Algorithms

- Many different ones, with different types
  - Multistep *vs*. single-step
  - Implicit *vs*. explicit
  - Step-size adjustment for better accuracy with fewer operations
  - Prefer higher-order algorithms
  - Special algorithms for stiff systems (wide variation in time constants)
  - Final could give new algorithm and ask you to take 2-3 steps with calculator

California State University Northridge 84