Programming and Numerical Integration

Larry Caretto
Mechanical Engineering 309

Numerical Analysis of Engineering Systems

April 2, 2014

California State University Northridge

Outline

- Schedule for remainder of term
- Review last lecture on regression
- Good programming practice guidelines
- Numerical evaluation of definite integrals (numerical quadrature)
  - Trapezoid rule
  - Simpson's Rule
  - Truncation errors for numerical quadrature
- Sixth programming assignment

California State University Northridge 2

Schedule for April and May

Monday	Tuesday	Wednesday	Thursday	Friday
	1	**2**	3	4
Spring Break				
14	15	**16 Quiz 7 & PA5**	17	16
21	22	**23**	24	35
28 PA 6	29	**30 Quiz 8**	*May 1*	2
5 PA 7	6	**7 Prog Exam**	8	9
12 Final				

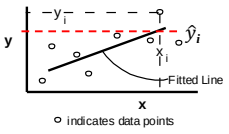
California State University Northridge 3

Review Regression/Interpolation

- Interpolation spasses an approximation function through every data point
  - Useful when we trust the data, *e.g.* finding intermediate values in data tables
  - Best approach is multiple, lower-order polynomials, like cubic splines
- Regression seeks an approximate relationship to experimental data
  - Relations between one dependent variable and one or more independent variable
  - Want to know uncertainty

California State University Northridge 4

Review 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

$$S = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}(y_i - a - bx_i)^2 \rightarrow \min$$

California State University Northridge 5

Regression Equations

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

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

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \qquad \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$s_{y|x} = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N-2}}$$

$$R^2 = 1 - \frac{(N-2)s_{y|x}^2}{\sum_{i=1}^{N} y_i^2 - N(\bar{y})^2} = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

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

t distribution

Standard errors for a and b

California State University Northridge 6

Student's t-Distribution Table

Critical t Distribution Values and Equivalents							
1-sided	80%	90%	95%	97.5%	99%	99.5%	99.9%
2-sided	60%	80%	90%	95%	98%	99%	99.8%
MATLAB	0.800	0.900	0.950	0.975	0.990	0.995	0.999
Excel	0.40	0.20	0.10	0.05	0.02	0.01	0.002
MathTable	0.2	0.1	0.05	0.025	0.01	0.005	0.001
$\nu = 1$	1.376	3.078	6.314	12.71	31.82	63.66	318.3
$\nu = 5$	0.920	1.476	2.015	2.571	3.365	4.032	5.893
$\nu = 10$	0.879	1.372	1.812	2.228	2.764	3.169	4.144
$\nu = 15$	0.866	1.341	1.753	2.131	2.602	2.947	3.733
$\nu = 30$	0.854	1.310	1.697	2.042	2.457	2.750	3.385
$\nu = 50$	0.849	1.299	1.676	2.009	2.403	2.678	3.261
$\nu = 100$	0.845	1.290	1.660	1.984	2.364	2.626	3.174
$\nu = 1000$	0.842	1.282	1.646	1.962	2.330	2.581	3.098

95% probability that $|t| \le 2.131$ and 97.5% probability that $t \le 2.131$ for $\nu = 15$

California State University Northridge

http://www.itl.nist.gov/div898/handbook/eda/section3/eda3672.htm

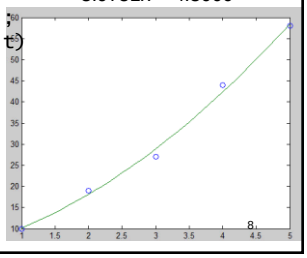
Review MATLAB 2nd-order polyfit

```
>> x=1:5;
>> y= [10 19   27   44   58];
>> p=polyfit(x,y,2)
p =   1.3571   3.9571   4.8000
>> xx = 1:.1:5;
>> yfit = polyval(p,xx);
>> plot (x,y,'o',xx,yfit)
```

Order of polynomial

polyfit returns coefficients p(k) such that p(x) = $p_1x^n + p_2x^{n-1} + \cdots + p_{n+1}$; here p(x) = $1.3571x^2 + 3.9751x + 4.8000$

polyval evaluates a polynomial with coefficients p_1 to p_{n+1} for input x, which is a row matrix here



California State University Northridge

Review Multilinear Regresion

- Have response variable, y that depends on predictive variables x_1, x_2, … x_K
  - Data sets y_m, x_{1m}, x_{2m}, … x_{Km} for m = 1, .. N
- *Equation for fitting one data set*

$$\hat{y}_m = b_0 + \sum_{j=1}^{K} b_j x_{jm}$$

- *Computing the R^2 value*

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

California State University Northridge

Review Excel LINEST Function

	A	B	C	D
1	y	x_1	x_2	x_3
2	2.55	3.00	440	500
3	1.95	3.47	350	400
4	1.89	3.14	440	540
5	2.24	3.46	350	370
6	2.31	3.59	450	480
7	1.74	1.75	200	320
8	1.87	3.03	310	470
9	0.83	3.18	290	400

Red letters and numbers indicate worksheet rows and colmns

- General formula is =LINEST(yRange, xRange, zero?, TRUE)
- Select K+1 columns and 5 rows for formula
- In this example =LINEST(A2:A9, B2:D9,, TRUE)
- Control+Shift+Enter
- Results on next slide
  - Intercept in last column

California State University Northridge

Review LINEST Results

	A	B	C
1	Slope K	Slope K – 1	Slope K – 2
2	Standard Error in Slope K(SE_K)	Standard Error in Slope (SE_{K-1})	Standard Erro (SE_{K-2})
3	R^2	$s_{y\|x}$	#N/A
4	F statistic	Degrees of Freedom	#N/A
5	Regression sum of squares	Residual sum of squares	#N/A

Rule of thumb: The slope and intercept should be at least twice their standard errors to be significantly different from zero

California State University Northridge

Review Example Results

	x_3 slope	x_2 Slope	x_1 Slope	Intercept
Value	-0.00509	0.00940	-0.5146	2.3966
Std err	0.00429	0.00448	0.4216	1.3503
R^2 | $s_{y\|x}$	0.579	0.444	#N/A	#N/A
F | df	1.835	4	#N/A	#N/A
SS	1.087	0.789	#N/A	#N/A

- F | df indicates F statistic and degrees of freedom in two adjacent columns
  - df = N – K – 1
- SS is sum of squares terms
- Bad fit because of large standard errors

California State University Northridge

Good Programming Practice

- Try to make programs as general as possible (within budget limits)
  - Instead of literal constants use consts (or variables) that can be readily changed
  - Plan for other possible calculation options, even if they are not implemented
  - Think about what ***you*** might like to have your program be able to do in the future
  - If working with others brainstorm ideas about what should be included

Planning Your Programs

- How will different components of the calculations be placed in different functions or subroutines
- Are there any repeated calculations that can be placed in one routine
- Design the overall composition of the program before doing any coding
- Can you use functions or subs from previous work
- Can anything that you do for this project be applied in later projects

Good Program Structure

- Code that is easy to read and understand
- Control flow that can be easily seen by someone reading the code
- Limit levels of nesting within a single function or sub
- Use a number of simple functions or subs
  - Each of which performs a small, distinct task
- Do not use goto statements
- A function/sub has only one entry point and should not have more than one exit point (exceptions sometimes used)

Use Comments Effectively

- Important to help another person to follow the code
  - May be yourself after some time
- Start each function/sub with comments
  - Give overall description of the function/sub, its purpose and general operation (references?)
  - Comment VBA DIM statements to tell readers the purpose of each variable.
  - Describe the inputs and outputs
  - Talk about algorithms used
    - Include revision history with names and dates
- Avoid obvious comments

Meaningful Variable Names

- Names that you can understand later or another person can read
  - Naming conventions for complex programs
- Use "title case": examples, beamStress, fluidPressure, staticTemperature, mainFlowArea
- Constants all CAPS in some conventions
- Can have simple names in small functions
  - Example on next slide
- Pick pattern for large number of variables such as combustorMainInletPressure

Simple Variable Name, s

```
Sub msgBoxArrayString( _
     s As String, x() As Double, _
     n As Long) As String

'

    Dim k As Long
    For k = 1 To n
        s = s & vbCrLf & "x(" & _
                  k & ") = " & x(k)
    Next k
    msgBox s
End Function
```

White Space Shows Structure

```
Sub test3(): Dim n
As Long: Dim x As
Double: Dim term
As Double: Dim sum
As Double: x = 1:
term = 1: sum =
term: For n = 1 To
10: term = term *
x / n: sum = sum +
term: Next n:
MsgBox sum:
Range("A1") = sum:
End Sub
```

```
Sub test3()
   Dim n As Long
   Dim x As Double
   Dim term As Double
   Dim sum As Double
   x = 1
   term = 1
   sum = term
   For n = 1 To 10
      term = term * x
      sum = sum + term
   Next n
   MsgBox sum
   Range("A1") = sum
End Sub
```


Good Programming

- Pretend that you are the user
- What would you like to see in a program
  - Easy to see what the program is supposed to do
  - A description of what each variable is
  - Comments that describe the code function
  - A structure that is easy to follow
    - Clear connection between For loop starts and corresponding Next statements, If statements and corresponding End If, *etc.*

Commenting Example

```
Function newton (x as double) as Variant

'Uses Newton's method to find root of the
'equation f(x) = 0.  (Note to write f(x)..
'Enter formula =newton(<initial guess>)
'<initial guess> may be cell reference,
'   literal value, or formula
'Root will be returned in function name
'For multiple roots, actual root found
'will depend on initial guess

    Const maxItr as Long = 100  'Comment
    Const maxRelErr as Double = 1e-14  'C
    Dim x2 As Double  'x value found by
    Dim k As Long    'Iteration counter
```


Commenting Example II

```
    Dim k As Long    'Iteration counter

'Apply Newton's method for maximum iter-
'ations and exit when solution is found.
'Return error if no solution after max i..
    For k = 1 To maxItr
        x2 = x - f(x) / fprime(x)
        If Abs(x - x2) <= maxRelError _
                          * Abs(x2) Then
            newton = x2 : Exit Function
        End If
    Next k
    newton = "ERROR"
End Function
```


Indenting Example

```
Function newton (x as double) as Variant
    Const maxItr as Long = 100  'Comment
    Const maxRelErr as Double = 1e-14  'C
    Dim x2 As Double  'x value found by
    Dim k As Long    'Iteration counter
    For k = 1 To maxItr
        x2 = x - f(x) / fprime(x)
        If Abs(x - x2) <= maxRelError _
                          * Abs(x2) Then
            newton = x2 : Exit Function
        End If
    Next k
    newton = "ERROR"
End Function
```


Numerical Integration

- Numerical integration is used for
  - Functions that have no analytic integral
    - We are able to compute f(x) for any x, but we cannot compute ∫f(x)dx analytically
  - Experimental or tabular data for which no functional relationship exists
- We integrate such functions by using interpolation between tabular data points or computed values of f(x)
  - See example at right

$$I = \int_{1.1}^{2.3} e^{-x^2} dx$$

Linear Polynomial Approximation

- Linear polynomial between x_1, $f(x_1) = f_1$, and x_2, $f(x_2) = f_2$ is $p_1(x) = f_1 + (f_2 - f_1)(x - x_1)/h$, where $h = x_2 - x_1$

$$I = \int_{x_1}^{x_2} f(x)dx \approx \int_{x_1}^{x_2} \left[f_1 + \frac{f_2 - f_1}{h}(x - x_1) \right] dx$$

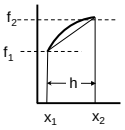
- Define $y = (x - x_1)/h$; $dy = dx/h$ so $dx = hdy$ and $y = 0$ at $x = x_1$ and $y = 1$ at $x = x_2$

$$\int_{x=x_1}^{x=x_2} \left[f_1 + \frac{f_2 - f_1}{h}(x - x_1) \right] dx = \int_{y=0}^{y=1} [f_1 + (f_2 - f_1)y] h dy$$

Linear Polynomial

$$\int_{x=x_1}^{x=x_2} \left[f_1 + \frac{f_2 - f_1}{h}(x - x_1) \right] dx = \int_{y=0}^{y=1} [f_1 + (f_2 - f_1)y] h dy =$$

$$h \left[f_1 y + (f_2 - f_1)\frac{y^2}{2} \right]_{y=0}^{y=1} = h \left[f_1 1 + (f_2 - f_1)\frac{1}{2} \right] = h \frac{f_2 + f_1}{2}$$



- Called the ***trapezoid rule*** because the area under the curve is approximated by the area of a trapezoid

Consider Larger Area

- Want to integrate f(x) from a to b
  - Subdivide interval into N segments with step size, h = (b – a)/N
    - Apply rule just found to each segment and add the results for all segments
    - Have $x_0 = a$, $x_k = x_0 + kh$ and at any x_k, define $f_k = f(x_k)$
    - From $x = x_0$ to $x = x_1$ we have $I_1 = h(f_1 + f_0)/2$
    - From $x = x_1$ to $x = x_2$ we have $I_2 = h(f_2 + f_1)/2$
    - From $x = x_2$ to $x = x_3$ we have $I_3 = h(f_3 + f_2)/2$
    - …
    - From $x = x_{N-1}$ to $x = x_N$, $I_N = h(f_N + f_{N-1})/2$

Pattern: each f_k appears twice except f_0 and f_N

Result for Larger Area

- Each value of the integrand, f_k, except for f_0 and f_N, appears twice. Adding all these terms gives the total as follows

$$T = h\frac{f_0 + f_1}{2} + h\frac{f_1 + f_2}{2} + h\frac{f_2 + f_3}{2} + \cdots + h\frac{f_{N-2} + f_{N-1}}{2} + h\frac{f_{N-1} + f_N}{2}$$

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

- Section 8.3.3 in Rao shows that the truncation error, E, for the Trapezoid rule, T, has a second-order error [$O(h^2)$]

Simpson's Rule

- Use a quadratic curve to approximate f(x) over an interval with 3 data points
- Consider x_0, $x_1 = x_0 + h$ and $x_2 = x_0 + 2h$
- Have $f_0 = f(x_0)$, $f_1 = f(x_1)$, and $f_2 = f(x_2)$
- Newton polynomial is $p_2(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$ with $a_0 = f_0$, $a_1 = (f_1 - f_0)/h$ and $a_2 = \{f_2 - [a_0 + a_1(x_2 - x_0)]\} / [(x_2 - x_0)(x_2 - x_1)] = \{f_2 - [f_0 + (f_1 - f_0)/h (2h)]\} / [(2h)(h)] = (f_2 - 2f_1 + f_0)/(2h^2)$

Simpson's Rule II

- Integrating the polynomial between x_0 and $x_2 = x_0 + 2h$ gives

$$I = \int_{x_0}^{x_2} f(x)dx \approx \int_{x_0}^{x_2} \left[f_0 + \frac{f_1 - f_0}{h}(x - x_0) + \frac{f_2 - 2f_1 + f_0}{2h^2}(x - x_1)(x - x_0) \right] dx$$

- Let $y = x - x_0$ so $dy = dx$ and $x - x_1 = x - (x_0 + h) = (x - x_0) - h = y - h$

$$\int_0^{2h} \left[f_0 + \frac{f_1 - f_0}{h} y + \frac{f_2 - 2f_1 + f_0}{2h^2}(y - h)y \right] dy =$$

$$\left[f_0 y + \frac{f_1 - f_0}{h}\frac{y^2}{2} + \frac{f_2 - 2f_1 + f_0}{2h^2}\left(\frac{y^3}{3} - \frac{hy^2}{2} \right) \right]_0^{2h}$$

Simpson's Rule III

- Apply integration limits

$$\left[f_0 y + \frac{f_1 - f_0}{h}\frac{y^2}{2} + \frac{f_2 - 2f_1 + f_0}{h^2}\left(\frac{y^3}{3} - \frac{hy^2}{2}\right)\right]_0^{2h} = 2hf_0 + \frac{f_1 - f_0}{h}\frac{4h^2}{2} + \frac{f_2 - 2f_1 + f_0}{2h^2}\left(\frac{8h^3}{3} - \frac{h4h^2}{2}\right) =$$

$$f_0\left(2h - 2h + \frac{h}{3}\right) + f_1\left(2h - \frac{2h}{3}\right) + f_2\frac{h}{3}$$

$$= \left(\frac{hf_0}{3} + \frac{4hf_1}{3} + \frac{hf_2}{3}\right) = \frac{h}{3}\left(f_0 + 4f_1 + f_2\right)$$

$$= \frac{1}{2h^2}\left(\frac{8h^3}{3} - \frac{h4h^2}{2}\right) = \frac{1}{2}\left(\frac{8h}{3} - \frac{4h}{2}\right) = \frac{1}{2}\frac{16h - 12h}{6} = \frac{1}{2}\frac{4h}{6} = \frac{h}{3}$$

$$I = \int_{x_0}^{x_2} f(x)dx = \int_{x_0}^{x_2} p(x)dx + E = \frac{h}{3}\left[f_0 + 4f_1 + f_2\right] + E$$

Simpson's Rule IV

- Apply to range from a to b with $x_0 = a$, $x_N = b$ and $h = (x_N - x_0)/N$ (Even N only)

$$S = \frac{h}{3}\left[f_0 + 4f_1 + f_2\right] + \frac{h}{3}\left[f_2 + 4f_3 + f_4\right] + \frac{h}{3}\left[f_4 + 4f_5 + f_6\right]$$

Even numbered terms appear twice in sum with factor of 1

Odd numbered terms appear once in sum with factor of 4

Other terms with f_6 to f_{N-4} go in here

Result

$$S = \frac{h}{3}\left[f_0 + f_N + 4\sum_{i=1,3,5}^{N-1} f_i + 2\sum_{i=2,4,6}^{N-2} f_i\right]$$

$$+ \frac{h}{3}\left[f_{N-4} + 4f_{N-3} + f_{N-2}\right] + \frac{h}{3}\left[f_{N-2} + 4f_{N-1} + f_N\right]$$

Simpson's Rule V

- See section 8.4.3 in Rao for derivation of truncation error for Simpson's rule

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

- Cutting h in half will reduce truncation error by a factor of 1/16
- Other Simpson's rules for different polynomials, but this is most popular
  - Sometimes called Simpson's one-third rule

Summary …to be continued

- Simpson's and Trapezoid

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

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

- Which method to use?
- How to choose step size, h?

$$h = \frac{b-a}{N}$$

$$x_k = a + kh$$

$$f_k = f\left(x_k\right)$$

MATLAB integral Function

- Numerical quadrature $\int_a^b f(x)dx$
- Basic form `>>integral(fun, a, b)`
  - fun is a function handle for integrand, f(x)
  - a and b are lower and upper limits
    - Uses global adaptive quadrature – varies the step size over the region [a b] to reduce global errors (errors at all parts of [a b])
  - Can also use additional arguments See MATLAB help for more information
  - Example: >>integral(@cos, 0, pi/2)

Sixth Programming Assignment

- Write a function in MATLAB and VBA to use Romberg integration for $\int_a^b f(x)dx$
  - Will give psuedocode in next lecture
  - Write same function in VBA and MATLAB
  - Use separate function for integrand, f(x)
    - Requires function handle in MATLAB
    - Use Application.Run(<function name as string>, <x value>) procedure in VBA
  - Apply to error function, erf(z) = $\int_0^z e^{-x^2}dx$
    - Compare to erf results in MATLAB and Excel

Due Monday, April 28 at 11:59 pm

Midterm Results

- Number of students 21
- Maximum score 105
- Average score: 75.3
- Median score: 73
- Standard deviation: 15.8
- Grade distribution:

```
52  53  54  59  63  64  66
66  70  71  73  74  76  79
86  90  93  93  98 100 101
```

California State University Northridge 37

Midterm Comments

- Trace loops step-by-step and select proper branches
  - Remember Mod as remainder function
- Order: $e_2/e_1 \approx (h_2/h_1)^n$; for $e_2/e_1 = 1/81$ with $h_2/h_1 = 1/3$, n = 4
- Multiplying an (n x 1) array by a (1 x m) array gives an (n x m) array
- Secant method needs two initial guesses, but they need not bracket root

California State University Northridge 38

Midterm Comments II

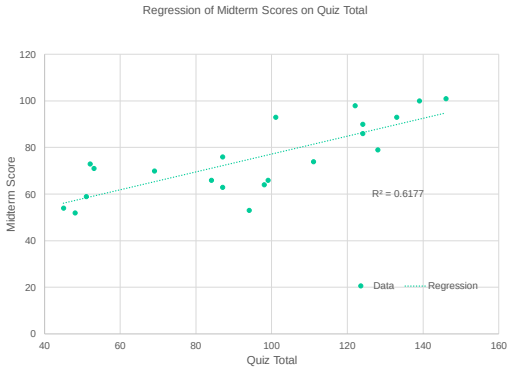
- Most problems with solving linear equations were due to simple errors
  - Could have done parts (a) and (b) as a single calculation
- For interpolation an n^{th}-order polynomial requires n+1 points as close as possible to the interpolation point
  - Write polynomial so that the interpolation point you are given is the independent variable: $c_p(T)$ for midterm problem

California State University Northridge 39

Midterm Comments III

- For finding roots of equations written as $f_1(x) = f_2(x)$ write equation as $f(x) = f_1(x) - f_2(x) = 0$ or $f(x) = f_2(x) - f_1(x)$
  - Can write sin(x) = cos(x) as f(x) = sin(x) – cos(x) = 0 or f(x) = cos(x) – sin(x) = 0, but not as tan(x) = 0
    - Could use tan(x) – 1 = 0
- Set calculator to radians for solving trig functions with other functions
  - Could have used degrees here, but initial guesses supplied assumed radians

California State University Northridge 40



California State University Northridge 41

Assignment Four Results

- Number of students 17
- Maximum score 40
- Average score: 24.6
- Median score: 28
- Standard deviation: 10.2
- Grade distribution:

```
 7  9  9 13 19 20
25 26 28 29 29
31 31 32 33 39 39
```

California State University Northridge 42

Assignment Four Comments

- Note that use of double *vs*. single decreases RMS error by about 10^8
- Pivoting decreases error by about 10
- MATLAB can get "infinite" solutions not possible with Excel or simple VBA
- Distinguish learning programming from learning tools in discussions
- Several errors in determinant programs
  - Need to retain pivoting for accuracy in computing upper triangular array

Assignment Four Comments II

- Be careful to pass variables to subs and functions where they are used
  - Can use module/global variables declared before first procedure to simplify process
    - Do not re-declare global variables in procedure
    - Initialize global variables to zero
      - They keep previous values otherwise!
- Use numerical values when discussing quantitative results
  - "the results had a relative difference of $1x10^{-8}$" instead of "the results were close."

Assignment Four Comments III

- In MATLAB, A(:) gives all columns of A as a single column array
- Could place determinant calculation in a function instead of a sub
  - Could also return function name from call to a Sub, i.e. can replace these two statements

```
Call detSub(A, det, N)  : detFct = det
```

  - With a single statement

```
Call detSub(A, detFct , N)
```


Assignment Four Comments IV

- Do not use term-by-term operation, A.*B, when you want matrix operations, A*B
- Can use Excel array formula for RMS: =SQRT(SUMSQ(A1:A9-B1:B9)/9)
  - Must press Control+Shift+Enter to construct an array formula, even if it is only for a value in one cell
    - Pressing only enter gives an error
  - Avoids need to compute individual differences in cells