Revision

Main topics:

- Looping (lecture 1)
- Numerical methods (lecture 3)
- Curve fitting (lecture 5)
- Interactive In and Output (lecture 7)

Revise the Task sheets, in particular:

Labsession 1
Labsession 3
Labsession 5
Labsession 7

1

Looping:

- Loops are mechanisms for repeating the same procedure
- Two structures in VBA for this: Do ... Loop and For ... Next
  - Do ... Loop is used when the loop terminates when a logical condition applies

```
Syntax: Do {While|Until} condition
            [statements]
        [Exit Do]
            [statements]
        Loop
```

  - In the DO WHILE ...LOOP the looping continues while the condition is true
  - In the DO UNTIL ...LOOP the looping continues until the condition is true
  - EXIT DO terminates the looping

2

- For ... Next is used when you know in advance how many times you want to iterate

Syntax: **For** counter = first **To** last [Step step]
[statements]
[**Exit For**]
[statements]
Next [counter]

- Exercises: Verify the following identities using looping:

(see Labsession 1 task 1)

$$\sum_{a=1}^{n}(2a-1)^5 = \tfrac{1}{3}n^2(16n^4-20n^2+7)$$

$$\sum_{a=1}^{n}(3-2a)^3 = n(2-n)(2n^2-4n+3)$$

3

- Example:

$$\sum_{a=1}^{n}(2a-1)^3 = n^2(2n^2-1)$$

```
Function LHS1(n)
 a = 1
 Do Until a = n + 1       (Do While a < n + 1)
   LHS1 = LHS1 + (2 * a - 1) ^3
   a = a + 1
 Loop
End  Function
Function LHS2(n)
 a = 1
 Do
  LHS2 = LHS2 + (2 * a - 1) ^3
  If a = n Then Exit Do
  a = a + 1
 Loop
End Function
```

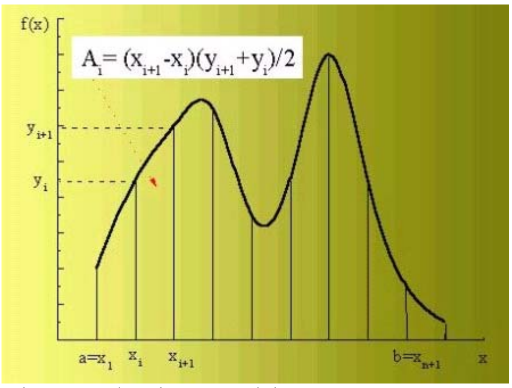
```
Function RHS(n)
   RHS = n ^2 * (2 * n ^2 - 1)
End Function
```

4

Numerical Methods with Excel/VBA:

➤ Numerical Integration

• Recall: $I = \int_a^b f(x)dx =$ area below the curve $f(x)$



- Idea: approximate the integral by sums over trapezoidal areas :

$$\boxed{I \approx \sum_{i=1}^{n} A_i}$$

5

- Take the subdivision of the domain [a,b] to be evenly spaced:

$$x_{i+1} - x_i = \frac{b-a}{n} = \Delta$$

$\Longrightarrow$ Trapezoid rule for integration:

$$\boxed{I \approx \Delta\left[\frac{1}{2}(y_1 + y_{n+1}) + \sum_{i=2}^{n} y_i)\right]}$$

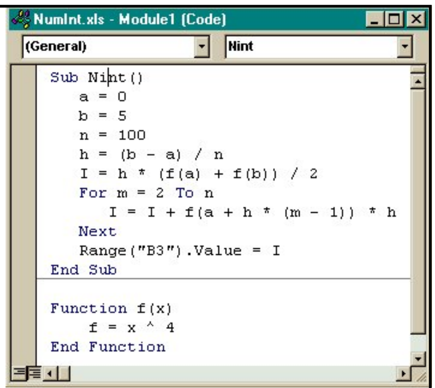
• Let us write a module (program) for this:

- Input: a ≡ lower integration limit
  b ≡ upper integration limit
  n ≡ number of subdivisions
  some function f(x) which we want to integrate

- Output: approximate value for the integral

6

```
Sub Nint()
  a = 0
  b = 5
  n = 100
  h = (b - a) / n
  I = h * (f(a) + f(b)) / 2
  For m = 2 To n
     I = I + f(a + h * (m - 1)) * h
  Next
  Range("B3").Value = I
End Sub
```



Put the result onto the Excel sheet into the cell B3

```
Function f(x)
   f = x ^ 4
End Function
```

7

- Example 1: $\int_0^5 x^4 dx = \frac{x^5}{5}\Big|_0^5 = 625$
  - The program gives:

$$n = 10 \qquad I = 635.4063$$
$$n = 100 \qquad I = 625.1042$$
$$n = 1000 \qquad I = 625.0010$$
$$n = 10000 \qquad I = 625.0000104$$

- Example 2: $\int_0^\pi \sin(x)dx = -\cos(x)|_0^\pi = 2$
  - Generate the π by 4 Arctan(1). In VBA this is written as 4 *Atn(1).
  - The program yields:

$$n = 10 \qquad I = 1.9835$$
$$n = 100 \qquad I = 1.999836$$
$$n = 1000 \qquad I = 1.999998$$

8

• Different types of methods:

- Simpson's 1/3 rule (based on three adjacent points):

$$I = \int_a^b f(x)dx \approx \tfrac{\Delta}{3}\left[\sum_{i=1,3,5,\ldots}^{n-2} y_i + 4y_{i+1} + y_{i+2} \right]$$

- Simpson's 3/8 rule (based on four adjacent points):

$$I = \int_a^b f(x)dx \approx \tfrac{3}{8}\Delta\left[\sum_{i=1,4,7,\ldots}^{n-3} y_i + 3y_{i+1} + 3y_{i+2} + y_{i+3} \right]$$

- Runge-Kutta methods, Monte Carlo integration,…
  · Here we do not derive these rules, but just take them as facts. See a different course on numerical methods for details.

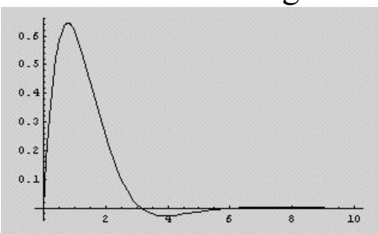
- Let us implement the Simpson's 3/8 rule as a user defined function.
- Implement the Simpson's 1/3 rule in Labsession 3.

9

```
Function Nintff(a, b, n)
  h = (b - a) / n
  I = 0
  For m = 1 To n - 2 Step 3
   I = I + (f(a+h*(m-1)) +3* f(a+h* m) +3* f(a+h*(m+1))+f(a+h*(m+2)))
  Next
  Nintff = I * h * 3 / 8
End Function
```

• Example 1: $2\int_0^\infty \sin(x)\exp(-x)dx = 1$

  · A problem here is to find a good cut-off for the upper limit.

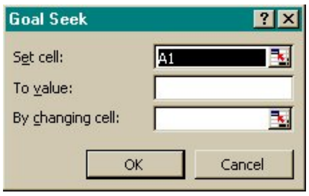


$\Longrightarrow$ b=10 ?

10

➤ Goal Seek

- Goal seek is a numerical routine implemented in Excel in form of a built-in function. It can be used to solve equations.
- Usage: select Tools → Goal Seek ↵ → a dialog window opens



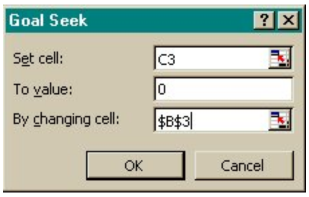
- Set cell contains the left hand side of an equation you want to solve
- To value contains the RHS of the equation
- By changing cell contains the variable of the equation

- Disadvantage: You have to guess a value near the answer.

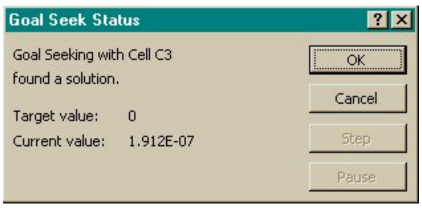
- Example: Solve the equation: $2x^2-9x-5=0$
  (We expect to find: $x_1=-1/2$ and $x_2=5$)

  - Type into the cell C3: =2*B3^2-9*B3-5
  - Type into the cell C4: =2*B4^2-9*B4-5

- Type into the cell B3 some starting value, e.g. -10
- open the Goal Seek dialog box and fill in



- OK ↵ →



- The cell B3 and C3 have changed to -0.5 and 1.912E-07
- Repeat this process for the cells C4 and B3 to find the other solution. (You need a new guess for the starting value.)

Curve Fitting

- On many occasions one has sets of ordered pairs of data $(x_1,\ldots,x_n,y_1,\ldots,y_n)$ which are related by a concrete function Y(X) e.g. some experimental data with a theoretical prediction

➤ suppose Y(X) is a linear function

$$Y = \alpha X + \beta$$

- Excel offers various ways to determine α and β

  i) SLOPE, INTERCEPT - functions based on the method of least square

$$\min = \sum_{i=1}^{n} [y_i - (\beta + \alpha x_i)]^2$$

SLOPE($y_1,\ldots,y_n, x_1,\ldots,x_n$,) $\rightarrow \alpha$

INTERCEPT($y_1,\ldots,y_n, x_1,\ldots,x_n$,) $\rightarrow \beta$

13

- How does Excel compute this? (see other courses for derivation)

· mean values: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

· slope: $\alpha = \sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y}) / \sum_{i=1}^{n}(x_i-\bar{x})^2$

· intercept: $\beta = \bar{y} - \alpha\bar{x}$

· regression coefficient:

$$r = \sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y}) / \sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2 \sum_{i=1}^{n}(y_i-\bar{y})^2}$$

A good linear correlation between the x_i and y_i -values is $r \cong 1$.

With VBA we can write a code which does the same job, see Lab-session 5 of Part II.

14

ii) LINEST - function

this function is more sophisticated than the previous one

LINEST(y_1,...,y_n,x_1, ...,x_n,*constant*,*statistics*)

- if *constant* = TRUE or omitted the intercept is computed otherwise it is zero
- if *statistics* = TRUE the function returns regression statistic values with the output:

slope	**intercept**
standard error in the slope	**standard error in the intercept**
r-squared	**standard error in the y estimation**

- we restrict ourselves here to

15

- notice that LINEST is an array function, such that you have to prepare for an output bigger than one cell:
  - · select a range for the output, e.g. 2×3 cells
  - · type the function, e.g. =LINEST(.....)
  - · complete with $\boxed{\text{Ctrl}}$ + $\boxed{\text{Shift}}$ + $\boxed{\text{Enter}}$

iii) adding a trendline

- this option also works for nonlinear, logarithmic, exponential ... correlations between the x- and y-values
  - · choose an XY-chart with the subtype which has no line
  - · right click any of the plotted points
    $\Rightarrow$ Add Trendline windows opens
  - · select the type of correlation, e.g. Linear, polynomial, ...
  - · in Options decide if you want to add the computed equation the r-squared value etc on the chart

16

Example:

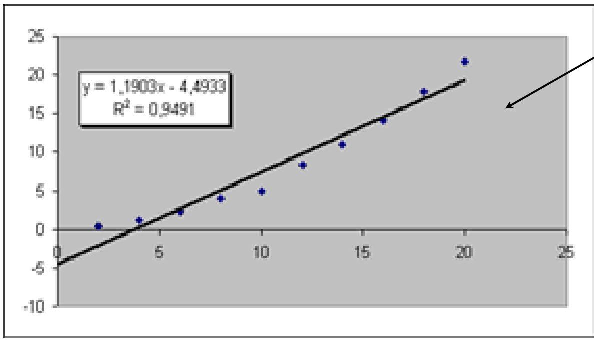
Consider the data:

2	0,4
4	1,2
6	2,3
8	4
10	5
12	8,3
14	11
16	14,1
18	17,9
20	21,8

assume linear correlation:

slope → 1.1903

intercept → -4,4933



with trendline adding

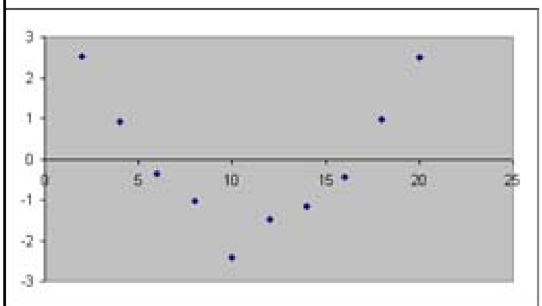
looks more or less linear?

17

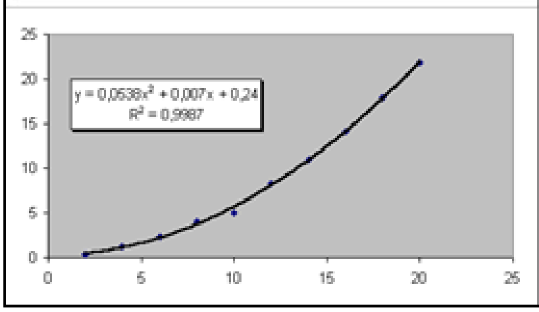
Compute the residuals, i.e. (the predicted values - the given ones):

$(1.1903\ x_i - 4.4933) - y_i$ →

2,512727
0,932121
-0,34848
-1,02909
-2,4097
-1,4903
-1,17091
-0,45152
0,967879
2,487273



not random!



quadratic fit is better!

18

Interactive In and Output

➤ Message box:

• displays a message in a dialog box and returns an integer value which depends on the answer of the user

syntax: return = MsgBox(prompt [, buttons] [, title])

· or:

syntax: return = MsgBox(prompt:= "..." , title:= " ..."] ...)

➤ Input box:

• displays a prompt in a dialog box, waits for the user to enter a text or click a button, and returns a string containing the content of the text box

syntax:
return = InputBox(prompt [,title] [,default] [,xpos] [,ypos])

19

➤ Exercises:

- (recall Labsession 7)
- Write a VBA code which simulates the following dialog:
  - · When executed the function should start with an Input Box which states "Did you finish your revision?". The title of this box should be "Revision".
  - · The entry into the input box should be assigned to a variable named "Answer". Declare the type of this variable as string.
  - · Design three message boxes with just an OK button both entitled "Revision". If the "Answer" is "Yes" the message box should say "Then do the revision test", if "Answer" is "No" then it should "You have time until the 11-th of May" otherwise "Answer with Yes or No!".

Good luck!

20