PETR 5313: CRN 38950, Fall 2017
Numerical Application in Petroleum Engineering, Lesson 01: Basic Calculation

Ekarit Panacharoensawad, PhD
Terry Fuller Petroleum Engineering Research Building Room 236
ekarit.panacharoensawad@ttu.edu

Cite as: Panacharoensawad, E. (2017) "Numerical application in petroleum engineering, Lesson 01: Basic Calculation", presentation slide for Texas Tech PETR 5313 Fall 2017

Outline

LibreOffice & Excel VBA

- Printing number to worksheet cells
- Write 1 to 1000
- Matrix operation
- Solver
- Coding practice: non-pythonic way of finding the prime number in the range of 0 to 2000. (VBA, LibreOffice)

Programming in Spreadsheet

LibreOffice is free in Ubuntu (free OS) in case you don't want to pay for Excel.

- Libre in French / Spanish means ‘free’

Programming Steps

- 1: Write Flow Chart (especially for the first time programming)
- 2: Break the whole tasks into sub-tasks
- 3: Code and test / debug each sub-tasks separately

VBA and OpenOffice.org Basic are very similar!

VBA Code can directly be used in OOBasic

  - Use `Option VBASupport 1`

OpenOffice.org (OOo) Basic Language Reference

If statement (branching)

- https://wiki.openoffice.org/wiki/Documentation/BASIC_Guide/Branching

Loop

- https://wiki.openoffice.org/wiki/Documentation/BASIC_Guide/Loops

Brief tutorial

- https://wiki.openoffice.org/wiki/Documentation/BASIC_Guide

More reference

- http://www.pitonyak.org/OOME_3_0.pdf

Objective

The objective of this lesson is for the learner to know how to use VBA and OpenOffice BASIC. In addition, the learner is expected to write similar program (See homework) that shown in the lecture.

The majority of the learning occur when you do the homework!

It may seem easy. You can actually tell, once you did it (not before).

Quick Review of Programming

Adding value of A by 1

```
A = A + 1
```

Summation over V(i) for every I

```
Sum = 0
For i = 1 to N
  Sum = Sum + V(i)
Next i
```

The code above is in VBA / OOo Basic

Quick Review of Programming…

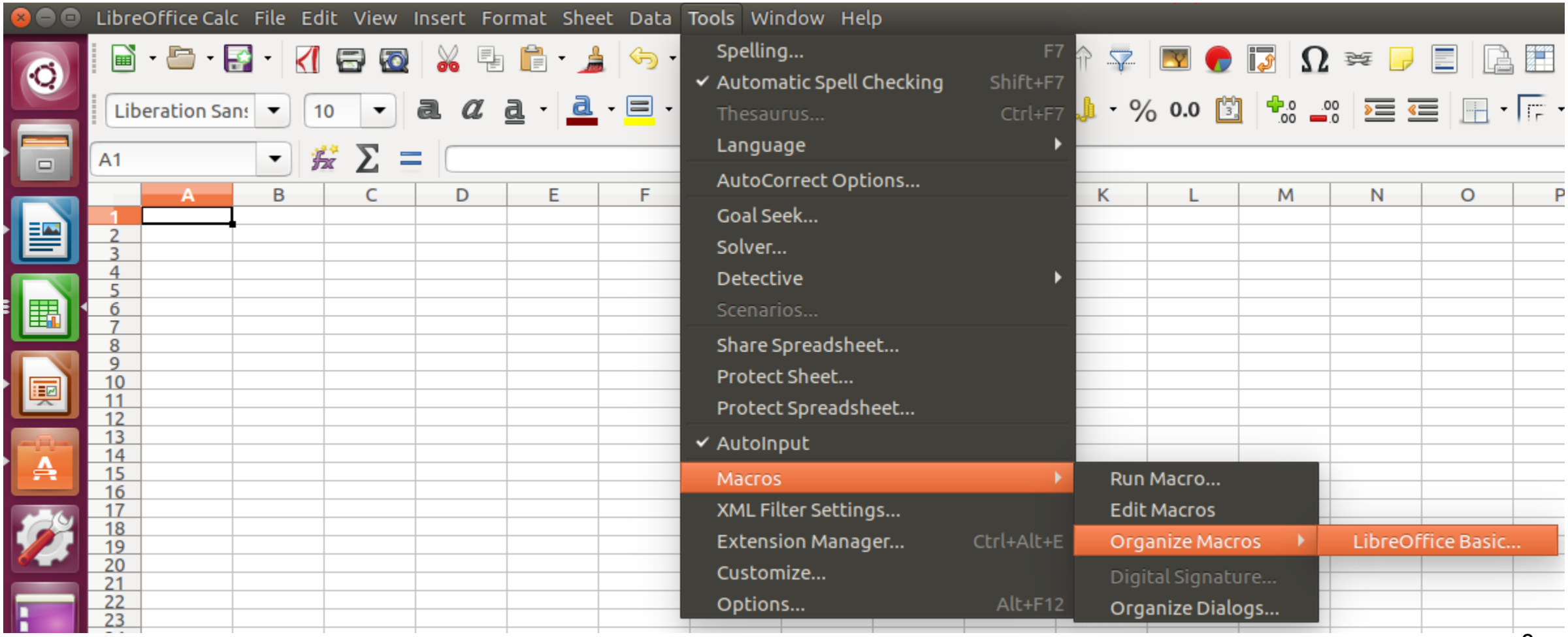
Swapping (= is the assignment operator)

```
A = 3 'now A = 3    B = ?   C = ?
B = 20  'now A = 3    B = 20    C = ?
C = A 'now A = 3    B = 20    C = 3
A = B 'now A = 20   B = 20    C = 3
B = C 'now A = 20   B = 3   C = 3
```

The code above is in VBA / OOo Basic

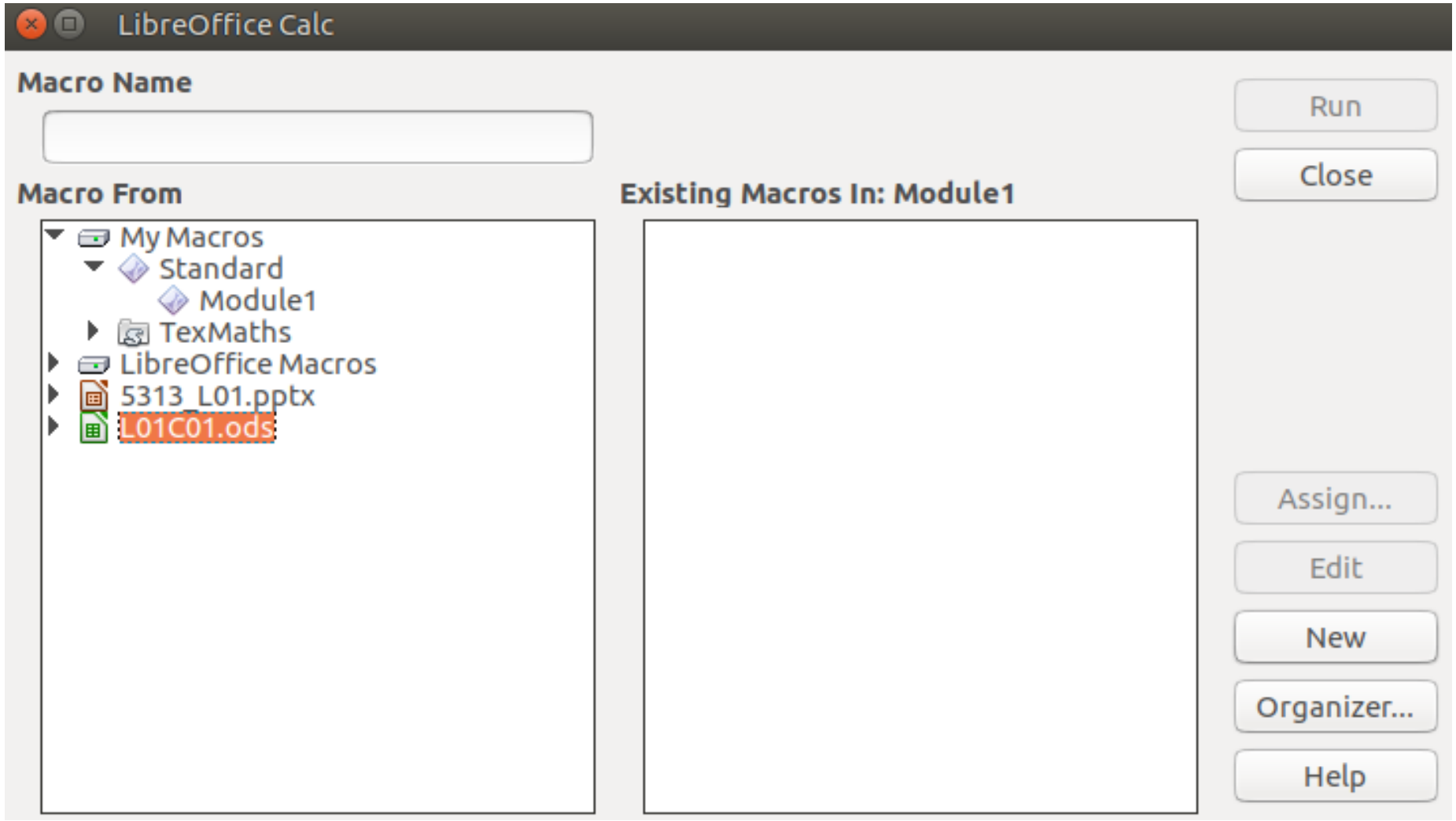
Lesson 01. Code 01 (L01C01)

- Create new macro in LibreOffice Calc (or alt+F11)



Creating Macro: LibreOffice Calc

- Click "New"



LibreOffice File Edit View Tools Window Help

[L01C01.ods].Standard

Object Catalog

- My Macros & Dial
  - Standard
    - Module1
  - TexMaths
- LibreOffice Macr
- L01C01.ods
  - Standard
    - Module1

```
REM  *****  BASIC  *****
Option Explicit

Sub Print_in_many_ways
    Dim Doc As Object
    Dim Sheet As Object
    Dim Cell As Object
    print "3"

    Doc = ThisComponent
    Sheet = ThisComponent.Sheets(0)
    Cell = Sheet.getCellByPosition(0,0)
    'write string to cell A1
    Cell.String = "hello world"

    'write number to cell B1
    Cell = Sheet.getCellByPosition(1,0)
    Cell.value = 1.235

    'write number to cell A2
    Cell = Sheet.getCellByPosition(0,1)
    Cell.value = 2.234

    'more info
    'help.libreoffice.org/Basic/Write_Statement_Runtime
    Dim ifile As Integer
    ifile = Freefile

    Cell = Sheet.getCellByPosition(0,2)
    Cell.string = CurDir

    Open "./dat.txt" For Output As ifile
    Write #ifile, "Hello ", 1.23, " ", 50
    Write #ifile, "World ", 1.23, " ", 51
    Close #ifile

End Sub
```

```
REM  *****  BASIC  *****
Option Explicit

Sub Print_in_many_ways
    Dim Doc As Object
    Dim Sheet As Object
    Dim Cell As Object
    print "3"

    Doc = ThisComponent
    Sheet = ThisComponent.Sheets(0)
    Cell = Sheet.getCellByPosition(0,0)
    'write string to cell A1
    Cell.String = "hello world"

    'write number to cell B1
    Cell = Sheet.getCellByPosition(1,0)
    Cell.value = 1.235
```

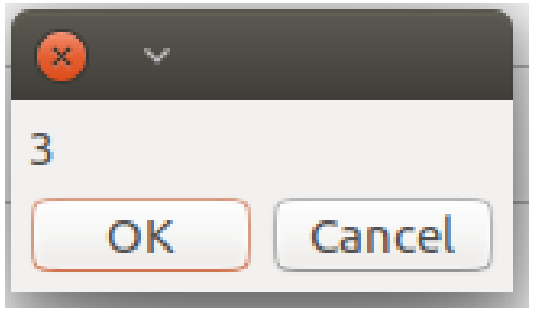
```
    'write number to cell A2
    Cell = Sheet.getCellByPosition(0,1)
    Cell.value = 2.234

    'more info
    'help.libreoffice.org/Basic/Write_Statement_Runtime
    Dim ifile As Integer
    ifile = Freefile

    Cell = Sheet.getCellByPosition(0,2)
    Cell.string = CurDir

    Open "./dat.txt" For Output As ifile
    Write #ifile, "Hello ", 1.23, " ", 50
    Write #ifile, "World ", 1.23, " ", 51
    Close #ifile

End Sub
```

L01C01 LibreOffice Calc Macro: Result

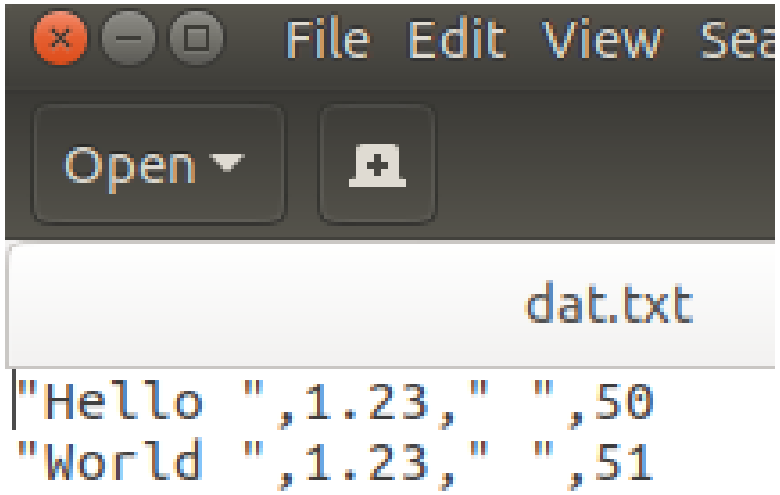
- Print to worksheet

	A	B
1	hello world	1.235
2	2.234	
3	/home/me	

MsgBox output (print)



- Print to file (print whatever we put in, line-by-line, character-by-character)



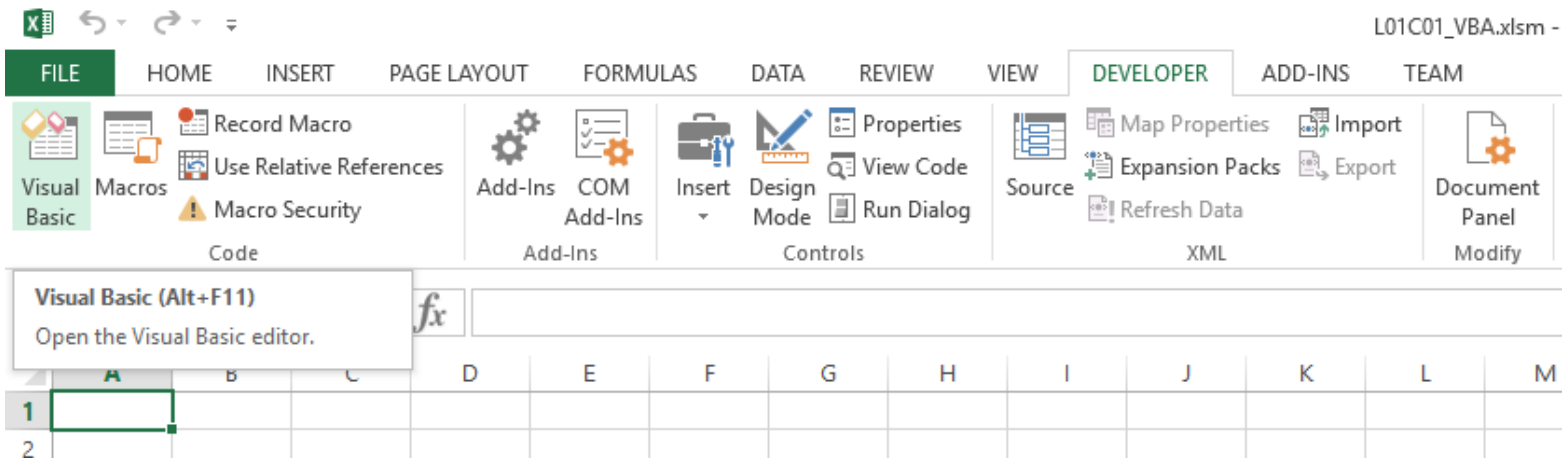
Quick Coding with VBA

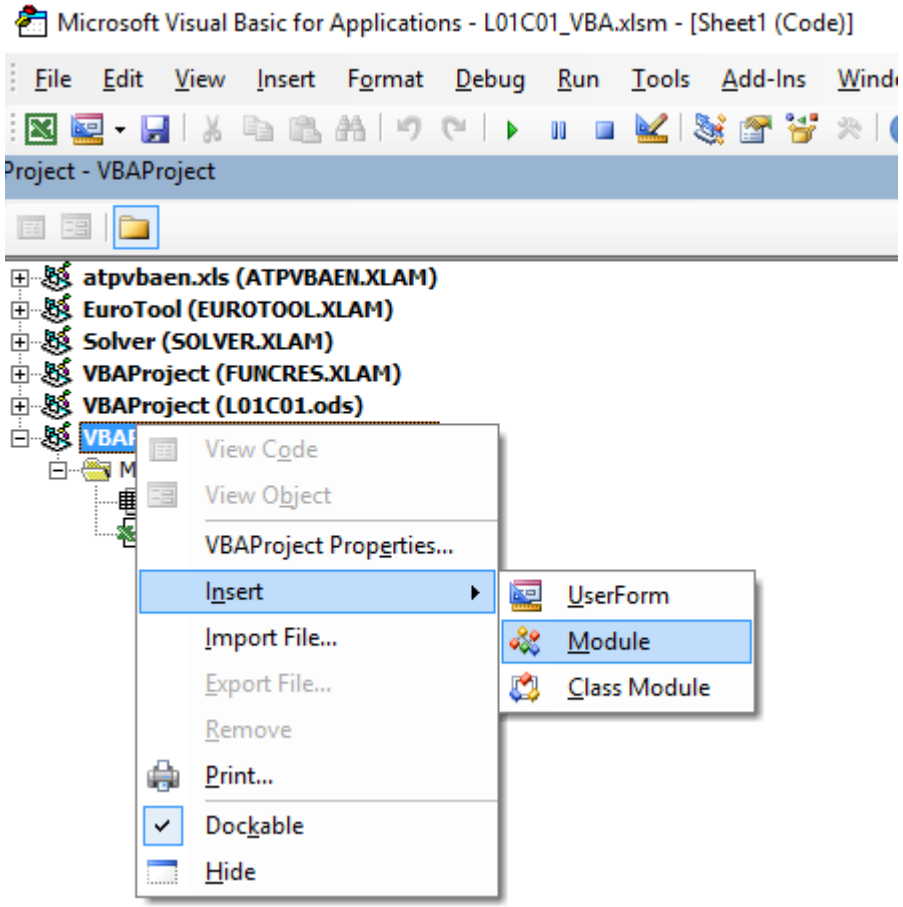
Access VBA screen

- Press Alt+F11
- Developer tab + click “Visual Basic”

Create Module

- Right click + insert + module





L01C01_VBA

```
Option Explicit

Sub L01C01()
    MsgBox ("3")
    Cells(1, 1) = "hello world"
    Cells(1, 2) = 1.235
    Cells(2, 1).Value = 2.234
    Cells(3, 1) = CurDir
    Dim ifile As Integer
    ifile = FreeFile
    'somehow, Linux file path work too!
    Open "./dat.txt" For Output As ifile
    Write #ifile, "Hello ", 1.23, " ", 50
    Write #ifile, "World ", 1.23, " ", 51
    Close #ifile
End Sub
```

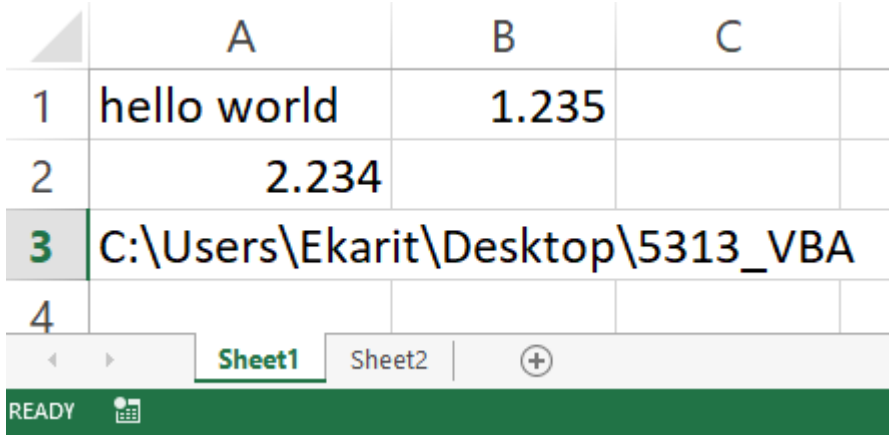
L01C01_VBA…

- Syntax are very similar to OOo Basic

Different parts

- `ThisComponent`
- VBA use (row, col), row/col start with 1
- OOo use (col, row), row/col start with zero
- A2 is `Cells(2, 1)`
- OOo can use `Print` directly
- VBA use `MsgBox`

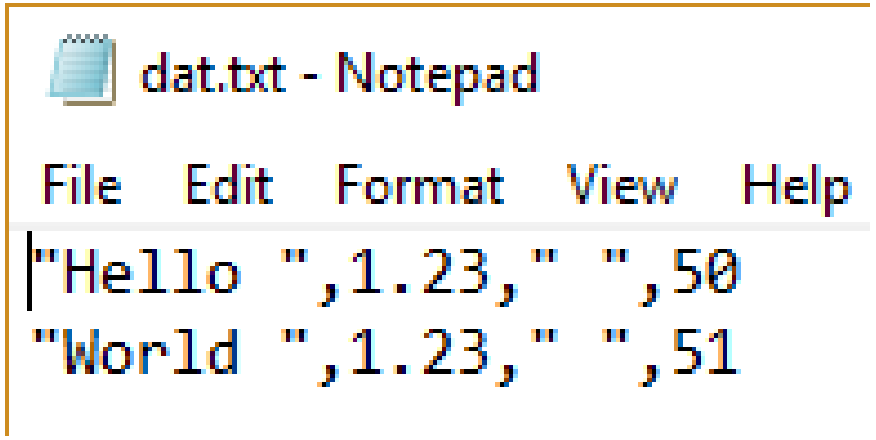
L01C01_VBA: Output



	A	B	C
1	hello world	1.235	
2	2.234		
3	C:\Users\Ekarit\Desktop\5313_VBA		
4			

Sheet1 Sheet2

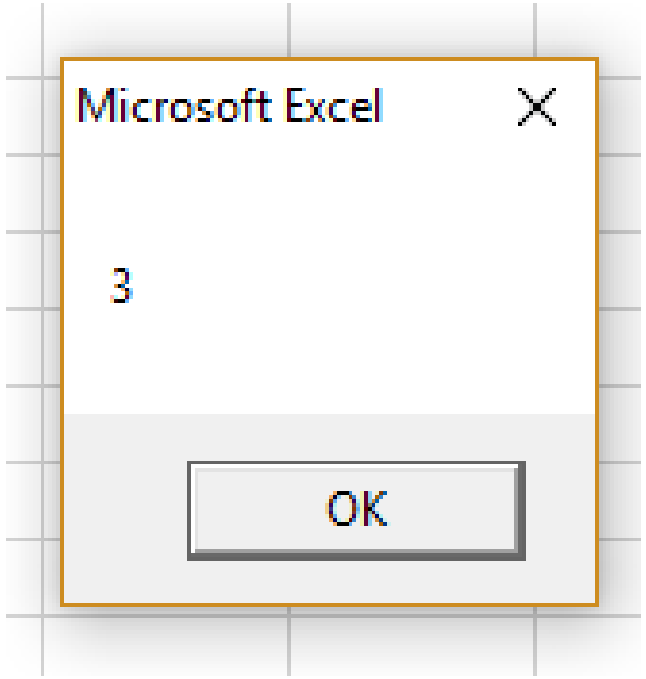
READY



dat.txt - Notepad

File Edit Format View Help

```
"Hello ",1.23," ",50
"World ",1.23," ",51
```



Microsoft Excel

3

OK

```
Sub L01C01_sheet2()
    'run from sheet 1. Need to create sheet2 first
    Sheets("Sheet2").Activate
    Cells(1, 1) = "hello world2"
    Cells(1, 2) = 1.2352
    Cells(2, 1).Value = 2.2342
    Cells(3, 1) = CurDir
    Dim ifile As Integer
    ifile = FreeFile
    Open "./dat2.txt" For Output As ifile
    Write #ifile, "Hello ", 1.23, " ", 502
    Write #ifile, "World ", 1.23, " ", 512
    Close #ifile
End Sub
```

L01C01_VBA: Write sheet 2

L01C01_VBA: Write sheet 2: Output

	A	B	C
1	hello world2	1.2352	
2	2.2342		
3	C:\Users\Ekarit\Desktop\5313_VBA		
4			

Sheet1 Sheet2

dat2.txt - Notepad

File Edit Format View Help

```
"Hello ",1.23," ",502
"World ",1.23," ",512
```

Run VBA on OpenOffice Basic Directly: L01XC01

Add

- Option VBASupport 1
- Option Compatible

Online conversion

- https://www.business-spreadsheets.com/vba2oo.asp

L01XC01
VBA to Basic

```
Rem Attribute VBA_ModuleType=VBAModule
Option VBASupport 1
Option Compatible
Option Explicit

Sub L01C01()
    MsgBox ("3")
    Cells(1, 1) = "hello world"
    Cells(1, 2) = 1.235
    Cells(2, 1).Value = 2.234
    Cells(3, 1) = CurDir
    Dim ifile As Integer
    ifile = FreeFile
    'somehow, Linux file path work too!
    Open "./dat.txt" For Output As ifile
    Write #ifile, "Hello ", 1.23, " ", 50
    Write #ifile, "World ", 1.23, " ", 51
    Close #ifile
End Sub
```

VBA Code on LibreOffice Calc

```
Sub L01C01_sheet2()
    'run from sheet 1. Need to create sheet2 first
    Sheets("Sheet2").Activate
    Cells(1, 1) = "hello world2"
    Cells(1, 2) = 1.2352
    Cells(2, 1).Value = 2.2342
    Cells(3, 1) = CurDir
    Dim ifile As Integer
    ifile = FreeFile
    Open "./dat2.txt" For Output As ifile
    Write #ifile, "Hello ", 1.23, " ", 502
    Write #ifile, "World ", 1.23, " ", 512
    Close #ifile
End Sub
```

L01C02_Print with for loop / if

- Objective 1: print 1 to 100, 10 number per row, 1 number in each cell
- Objective 2: in sheet 2, print 30 to 60 in column 1 and check if each number is a prime number or not
- Objective 3: in sheet 3, print all prime number to cell in column 1, for the number in the range of 1 to 20000
  - Print the last 10 prime number in column 2 in the descending order
  - How many prime number do we have in this range?
    - Print this value to "D2"

L01C02_Objective 1: Approach

- Use nested for loop
- Write from left to right, first. Then, writing down.
- Let's define
  - The index for row as i
  - The index for column as j
- Let's have
  - i = 1
  - j goes from 1 to 10
  - i = 2
  - j goes from 1 to 10 (Again!)

L01C02_Objective 1: One way to think

Nested for loop give

- i = 1, j = 1, 2, 3, ..., 10
- i = 2, j = 1, 2, 3, ..., 10
- ...
- i = 10, j = 1, 2, 3, ..., 10

Value output:

- out = j (for i = 1, is OK only for the 1st row)
- out = j + 10 (for i = 2, is OK only for the 2nd row)
- out = j + 20 (for i = 3, is OK only for the 3rd row)
- out = j + (i - 1) * 10, (is OK for ith row)

L01C02_Objective 1: Pseudo code

```
For i = 1 to 10
    For j = 1 to 10
        out = j + (i - 1) * 10
        print out to the cell row i, column j
    Next j
Next i
```

L01C02_Objective 1: Code

```
Sub Objective1
    Dim i as integer
    Dim j as integer
    Dim Doc as Object
    Dim Sheet as Object
    Doc = ThisComponent
    Sheet = Doc.Sheets.getByName("Sheet1")
    for i = 1 to 10
        for j = 1 to 10
            Cell = Sheet.getCellByPosition(j, i)
            Cell.value = j + 10 * (i-1)
        next j
    next i
End Sub
```

L01C02_Objective 1: Result

	A	B	C	D	E	F	G	H	I	J	K
1											
2		1	2	3	4	5	6	7	8	9	10
3		11	12	13	14	15	16	17	18	19	20
4		21	22	23	24	25	26	27	28	29	30
5		31	32	33	34	35	36	37	38	39	40
6		41	42	43	44	45	46	47	48	49	50
7		51	52	53	54	55	56	57	58	59	60
8		61	62	63	64	65	66	67	68	69	70
9		71	72	73	74	75	76	77	78	79	80
10		81	82	83	84	85	86	87	88	89	90
11		91	92	93	94	95	96	97	98	99	100

L01C02_Objective 1: VBA Code

```
Option Explicit
Sub Objective1() 'VBA
    Dim i As Integer
    Dim j As Integer
    Sheets("Sheet1").Activate
    For i = 1 To 10
        For j = 1 To 10
            Cells(i + 1, j + 1) = j + 10 * (i - 1)
        Next j
    Next i
End Sub
```

L01C02_Objective 1: VBA Result

	A	B	C	D	E	F	G	H	I	J	K
1											
2		1	2	3	4	5	6	7	8	9	10
3		11	12	13	14	15	16	17	18	19	20
4		21	22	23	24	25	26	27	28	29	30
5		31	32	33	34	35	36	37	38	39	40
6		41	42	43	44	45	46	47	48	49	50
7		51	52	53	54	55	56	57	58	59	60
8		61	62	63	64	65	66	67	68	69	70
9		71	72	73	74	75	76	77	78	79	80
10		81	82	83	84	85	86	87	88	89	90
11		91	92	93	94	95	96	97	98	99	100

Possible Quiz1 (10 minutes)

Hand-Writing-Code

Write (by hand, on paper) either VBA code or OpenOffice.org Basic (OOo Basic) code for

- Printing i ^ 0.5 for i = 1, 2, 3, … , 100 onto worksheet
- Each row contain 10 numbers / print 10 rows totally
- One number per cell
- Need to use for loop
- Write `Option Explicit` as the first statement
- Write the value not string (9.53939 not “91^0.5”)
- You may want to practice serval times to memorize it

L01C02_O2: Check if prime: Approach

- Input range is 30 to 60

From 0 to 60, prime numbers are Solution (From google)

- 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61
- Even number is not a prime number, except 2
  - Decrease the domain by half
- The positive divisor of a prime number is itself and one.

We need to know the prime number below 30 too, in order to tell if the number above 30 is a prime number or not

L01C02_O2: Thinking Approach

Solution: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61

- Let's use for loop with if statement

Conditions

- In the range of 30 to 60, if the number is an even number, it is not a prime number
- Otherwise, check if it is divisable by any number below it
- Use MOD (modulo to find the remainder after division)

L01C02_O2
Thinking Out Loudly

- Is 31 a prime number or not?
- 31 MOD 2 = 1
- 31 MOD 3 = 1
- 31 MOD 4 = 3
- ...
- MOD result is always greater than 0
- Do we need to check all # < 31?
  - No, we can check just prime # <31.
  - This will be a lot quicker!

#1	#2	mod	#1	#2	mod
31	1	0	31	17	14
31	2	1	31	18	13
31	3	1	31	19	12
31	4	3	31	20	11
31	5	1	31	21	10
31	6	1	31	22	9
31	7	3	31	23	8
31	8	7	31	24	7
31	9	4	31	25	6
31	10	1	31	26	5
31	11	9	31	27	4
31	12	7	31	28	3
31	13	5	31	29	2
31	14	3	31	30	1
31	15	1	31	31	0
31	16	15			

L01C02_O2: Steps

1. Get the value to be checked (x) if it is a prime number or not
2. Get the list of the prime number for the range of [2, x-1]
3. Do MOD(x, p) where p is in the list from the step 2
    1. Whenever MOD = 0, exit the loop (break)
    2. If MOD <> 0, keep on doing
    3. If MOD <> 0, for all, then x is the prime number

Note: Try to avoid a recursive algorithm

- Difficult to debug
- Can be an exponential time complexity if not done correctly
- Step 2 requires the previous call of step 2 that is not exist

L01C02_O2: Revised procedure

Creating the list of prime number from 1 to 60 then select the values that are >= 30 and <= 60

1. Assume the known prime number to be [2, 3]
2. Start with 5
3. Check all prime number less than 5 (which is 2 & 3)
    1. If no MOD(y, n) = 0 for n = 2, 3, ... , y-1
    2. If any MOD(y, n) = 0, y is not the prime number
4. Either keep y as prime # or discard, then move to the next number

L01C02_O2: Step by step

For only odd numbers starting from 5

Check #5

- Check MOD(5,3)
  - add #5 to the list

Check #7

- Check MOD(7,3)
- Check MOD(7,5)
  - add #7 to the list

Check #9

- Check MOD(9,3)
  - MOD = 0

Check #11

- Check MOD(11,3) ...
- Keep checking until 60

L01C02_O2: Data Structure

Adding value to list

1. `Dim p_list(1 to 5) as integer 'including 1 & 5`
2. Typically, **array starts from zero**
3. If the list is full, double the capacity
   - `Redim preserve p_list(1 to n*2) as integer`

Check if the list is full

- `Ubound(p_list)` to get the upper bound
- Compare the Upper bound value with the number of the prime numbers that are found

L01C02_O2: Code part 1/3

```
Sub Objective2
    Dim Doc as Object
    Dim Sheet as Object
    Dim p_list(1 to 5) as integer
    Dim N as Integer
    Dim x as Integer 'upper limit for p# search
    Dim val as Integer 'value to be checked
    Dim i as Integer 'index
    Dim up_val as Integer
    x = 60
    p_list(1) = 2
    p_list(2) = 3
    N = 2
```

```
For val = 5 to x Step 2 'check if val be p#
    For i = 1 to N 'check against the list of p#
        If val MOD p_list(i) = 0 Then
            Exit For
        End If
        If i = N Then 'not exit means p# is found
            up_val = UBound(p_list)
            If up_val > N Then 'check if the space
                p_list(N+1) = val 'is large enough
                N = N + 1 'update N (adding 1)
            Else
                Redim Preserve p_list(N*2) as Integer
                p_list(N+1) = val
                N = N + 1 'update N (adding 1)
            End If
        End If
    Next i
Next val
```

L01C02_O2: Code part 2/3

```
    Doc = ThisComponent
    Sheet = Doc.Sheets.getByName("Sheet2")
    Cell = Sheet.getCellByPosition(7, 0)
    Cell.string = "all answer"
    for i = 1 to N 'print all answers
        Cell = Sheet.getCellByPosition(7, i)
        Cell.value = p_list(i)
    next i
    Cell = Sheet.getCellByPosition(8, 0)
    Cell.string = "needed answer"
    Dim j as Integer 'printing index
    j = 1
    for i = 1 to N 'print the value if 30 <= x <= 60
        Cell = Sheet.getCellByPosition(8, j)
        If (p_list(i) >= 30) and (p_list(i) <= 60) Then
            Cell.value = p_list(i)
            j = j + 1
        End If
    next i
End Sub
```

L01C02_O2: Code part 3/3

L01C02_O2: Result

	H	I
1	all answer	needed answer
2	2	31
3	3	37
4	5	41
5	7	43
6	11	47
7	13	53
8	17	59
9	19	
10	23	
11	29	
12	31	
13	37	
14	41	
15	43	
16	47	
17	53	
18	59	

Equivalent result from VBA code

	A	B	C
1	all answer	needed answer	
2	2	31	
3	3	37	
4	5	41	
5	7	43	
6	11	47	
7	13	53	
8	17	59	
9	19		
10	23		
11	29		
12	31		
13	37		
14	41		
15	43		
16	47		
17	53		
18	59		

L01C02_O2: VBA Code part 1/3

- Notice that p_list was declared with `ReDim` not `Dim`

```
Sub L01C02_O2_VBA()
    ReDim p_list(1 To 5) As Integer
    Dim N As Integer
    Dim x As Integer 'upper limit for p# search
    Dim val As Integer 'value to be checked
    Dim i As Integer 'index
    Dim up_val As Integer
    x = 60
    p_list(1) = 2
    p_list(2) = 3
    N = 2
```

L01C02_O2: VBA Code part 2/3

```
For val = 5 To x Step 2 'check if val be p#
    For i = 1 To N 'check against the list of p#
        If val Mod p_list(i) = 0 Then
            Exit For
        End If
        If i = N Then 'not exit means p# is found
            up_val = UBound(p_list)
            If up_val > N Then 'check if the space
                p_list(N + 1) = val 'is large enough
                N = N + 1 'update N (adding 1)
            Else
                ReDim Preserve p_list(1 To N * 2) As Integer
                p_list(N + 1) = val
                N = N + 1 'update N (adding 1)
            End If
        End If
    Next i
Next val
```

L01C02_O2: VBA Code part 3/3

```
    Cells(1, 1).Value = "all answer"
    For i = 1 To N 'print all answers
        Cells(i + 1, 1) = p_list(i)
    Next i
    Cells(1, 2) = "needed answer"
    Dim j As Integer 'printing index
    j = 1
    For i = 1 To N 'print the value if 30 <= x <= 60
        If (p_list(i) >= 30) And (p_list(i) <= 60) Then
            Cells(j + 1, 2) = p_list(i)
            j = j + 1
        End If
    Next i
End Sub
```

VBA VS OpenOffice.org Basic

- VBA is not free, but OOoB is free
- VBA capitalize letter & adjust spacing automatically (OOoB doesn't)
- Some different in row column reference to cell
- VBA use (row, col) while OOoB use (col, row)
- VBA & OOoB both can `ReDim Preserve`
- Array declare with a specified size `Dim A(5) as Long`, cannot be `ReDim` (in VBA, but can be `ReDim` in OOo Basic)
- WPS Office for Linux/Ubuntu (paid version) has VBA and very close to Excel (free version does not have VBA)

L01C02_O3: Do it later as a homework

Hint

- Use code from C02O2
- Use `step -1` to move backward in for loop
- You just need to change the L01C02_O2 code a little, then it should work (if you do it with LibreOffice Calc).
- For doing it in VBA, see the next couple of slides

It is my intention to put the code in L01C02_O2 in a picture format, so that you have an opportunity to practice.

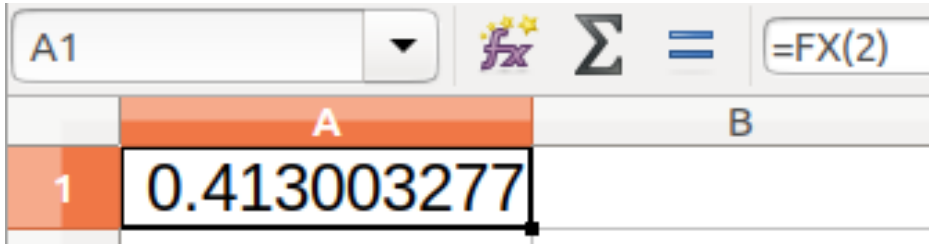
L01C02_O3: Expected Result

- Note: Column 1 has many more number

	A	B	C	D
1	all answer	needed answer		total p#
2	2	19997		2262
3	3	19993		
4	5	19991		
5	7	19979		
6	11	19973		
7	13	19963		
8	17	19961		
9	19	19949		
10	23	19937		
11	29	19927		
12	31			
13	37			
14	41			
15	43			
16	47			
17	53			

User-Defined Function: OOoB

- $f(x) = 0.75 * (1 - exp(-0.4 * x))$




```
REM  *****  BASIC  *****
Option Explicit
Function fx(x As Double) As Double
    fx = 0.75 * (1 - Exp(-0.4 * x))
End Function
```

- `x As Double` means, `fx` takes input as `Double`
- `fx( ) as Double`, means, output or `fx` is `Double`
- If the function in Basic is change, press Ctrl + Shift + F9 to update every cells

User-Defined Function: VBA

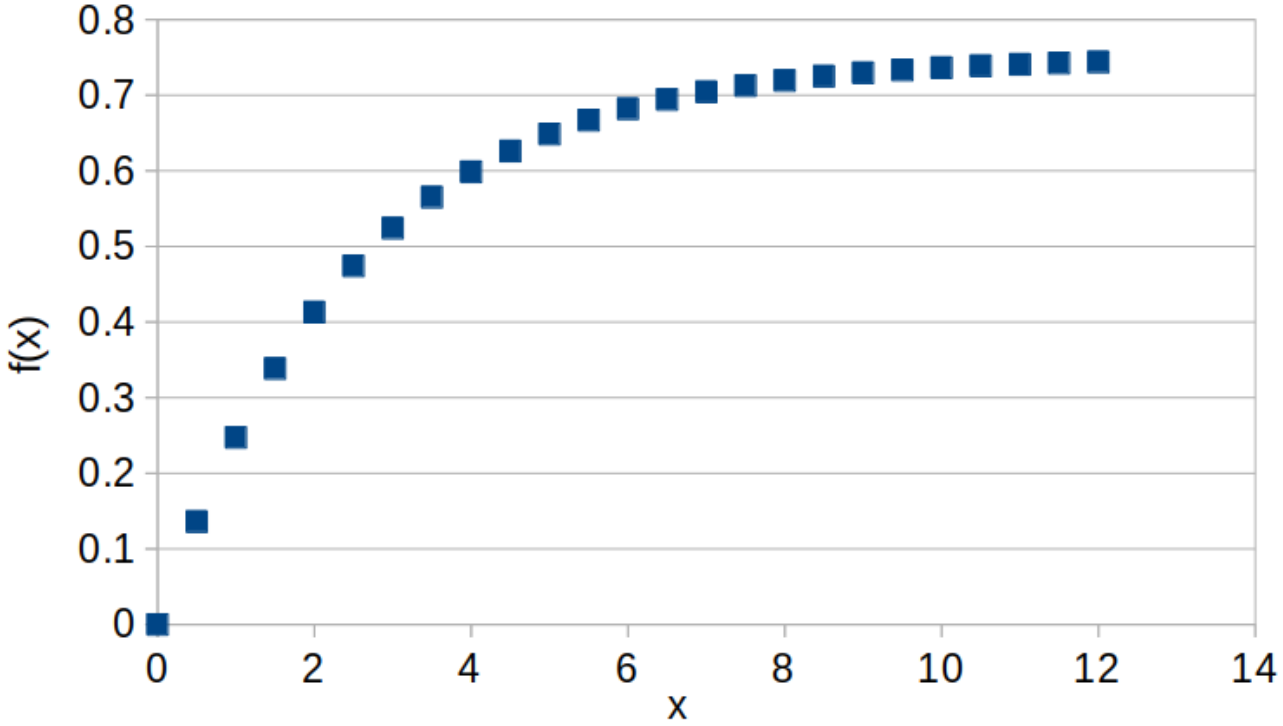
- f(x) = 0.75 * (1 – exp(-0. 4 * x))

```
Function fx(x As Double) As Double
    fx = 0.75 * (1 - Exp(-0.4 * x))
End Function
```

- Function call from Excel worksheet
- `=fx(A10)`
  - Put value from cell "`A10`" as the value of "`x`" in function `fx`
- Function can also be called from VBA subroutine

Solver: L01C03 (both Excel and Calc)

- Non-linear solver can be used to find unknown constants in non-linear equation
- Example: Data from non-linear function f(x) = 0.75 * (1 – exp(-0.4 * x)) are generated. For f(x) = A * (1 – exp(- B * x)), solver can find A and B that are allow the best fit with the data easily



Solver: Step-By-Step (both Excel & Calc)

Create

- Column A: x values from 0 to 12, at the interval of 0.5
- Column B: f(x) true values
- Column C: f(x) true values created from VBA function (just for practice)
- Column D: f(x) predicted values
  - based on the initial guess of A = 1 and B = 1
- Column E: (true – predict) ^ 2

Solver: Step-By-Step…

1 Actual input (ctrl + ` or ~) (Excel)

2 Normal view

3 Graphs

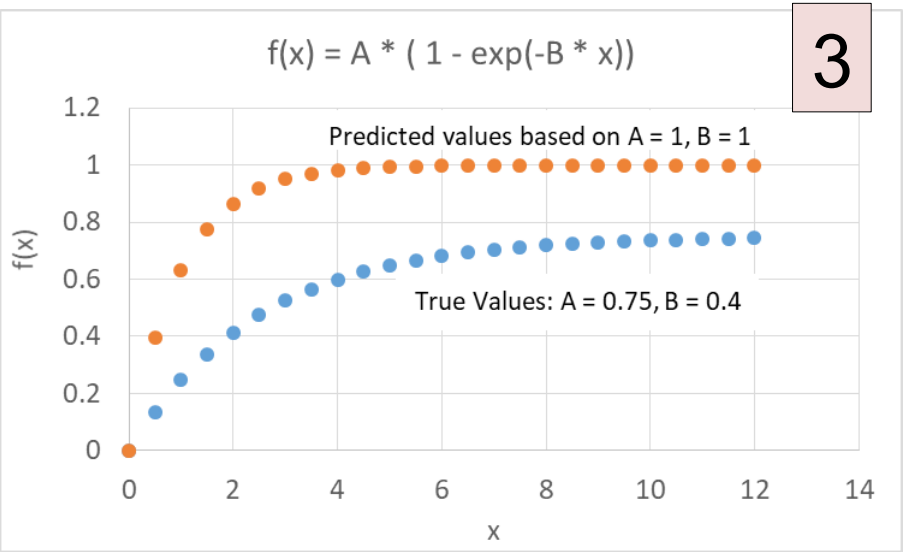
1

	A	B	C	D	E
1	A	1		initial val	1
2	B	1			1
3	f(x) = A * (1 - exp(-B * x))				
4	SSE	=SUM(E9:E33)			
5					
6					
7		True val	True val		
8	x	fx (excel)	fx (VBA)	predict	err^2
9	0	=0.75 * (1 - EXP(-0.4	=fx(A9)	=$B$1*(1-EXP(-$B$2*A9))	=(D9-B9)^2
10	=A9+0.5	=0.75 * (1 - EXP(-0.4	=fx(A10)	=$B$1*(1-EXP(-$B$2*A10))	=(D10-B10)^2
11	=A10+0.5	=0.75 * (1 - EXP(-0.4	=fx(A11)	=$B$1*(1-EXP(-$B$2*A11))	=(D11-B11)^2
12	=A11+0.5	=0.75 * (1 - EXP(-0.4	=fx(A12)	=$B$1*(1-EXP(-$B$2*A12))	=(D12-B12)^2
13	=A12+0.5	=0.75 * (1 - EXP(-0.4	=fx(A13)	=$B$1*(1-EXP(-$B$2*A13))	=(D13-B13)^2
14	=A13+0.5	=0.75 * (1 - EXP(-0.4	=fx(A14)	=$B$1*(1-EXP(-$B$2*A14))	=(D14-B14)^2
15	=A14+0.5	=0.75 * (1 - EXP(-0.4	=fx(A15)	=$B$1*(1-EXP(-$B$2*A15))	=(D15-B15)^2

2

	A	B	C	D	E
1	A	1		initial val	1
2	B	1			1
3	f(x) = A * (1 - exp(-B * x))				
4	SSE	2.65501			
5					
6					
7		True val	True val		
8	x	fx (excel)	fx (VBA)	predict	err^2
9	0	0	0	0	0
10	0.5	0.13595	0.13595	0.39347	0.06632
11	1	0.24726	0.24726	0.63212	0.14812
12	1.5	0.33839	0.33839	0.77687	0.19226
13	2	0.413	0.413	0.86466	0.204
14	2.5	0.47409	0.47409	0.91792	0.19698

3

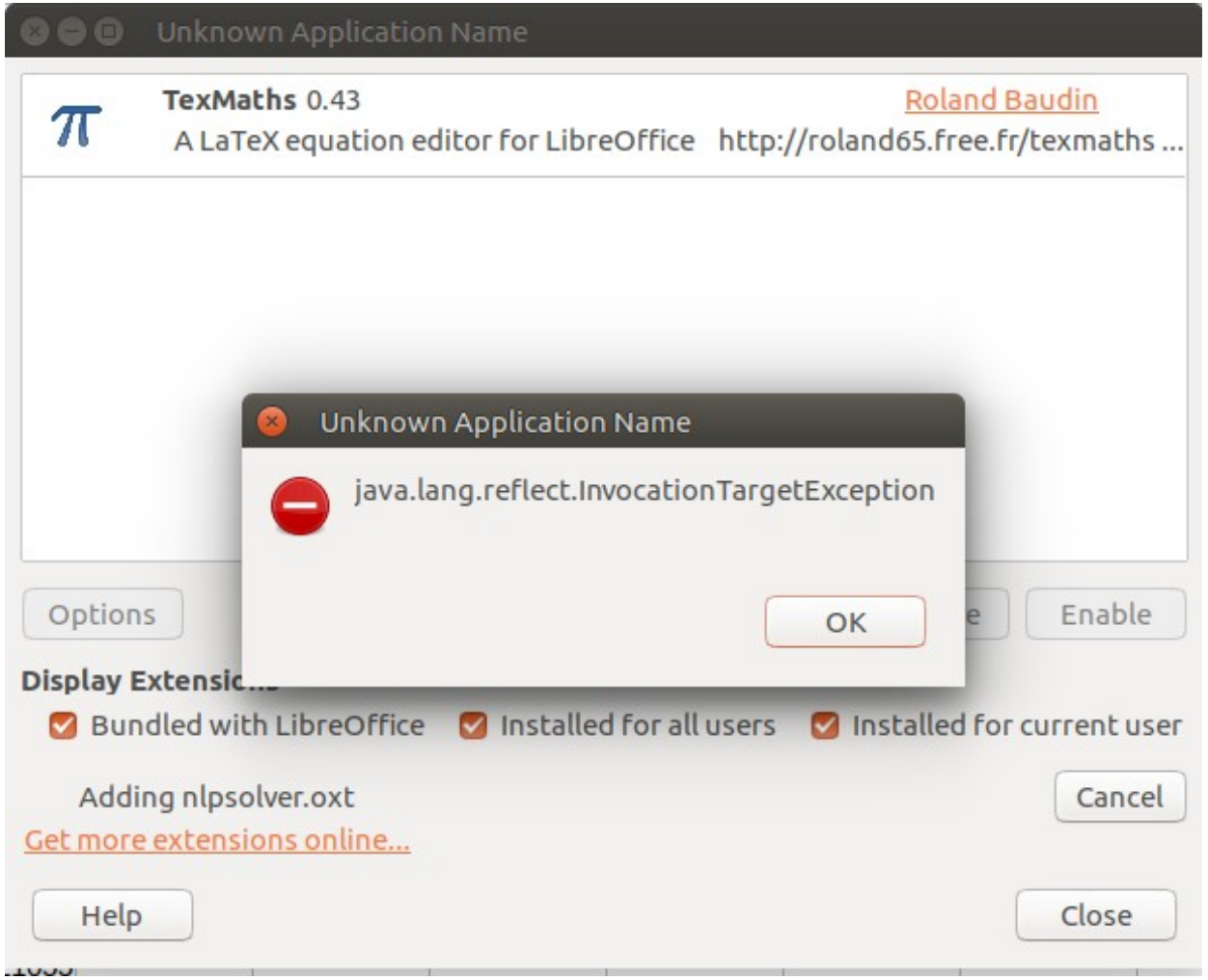


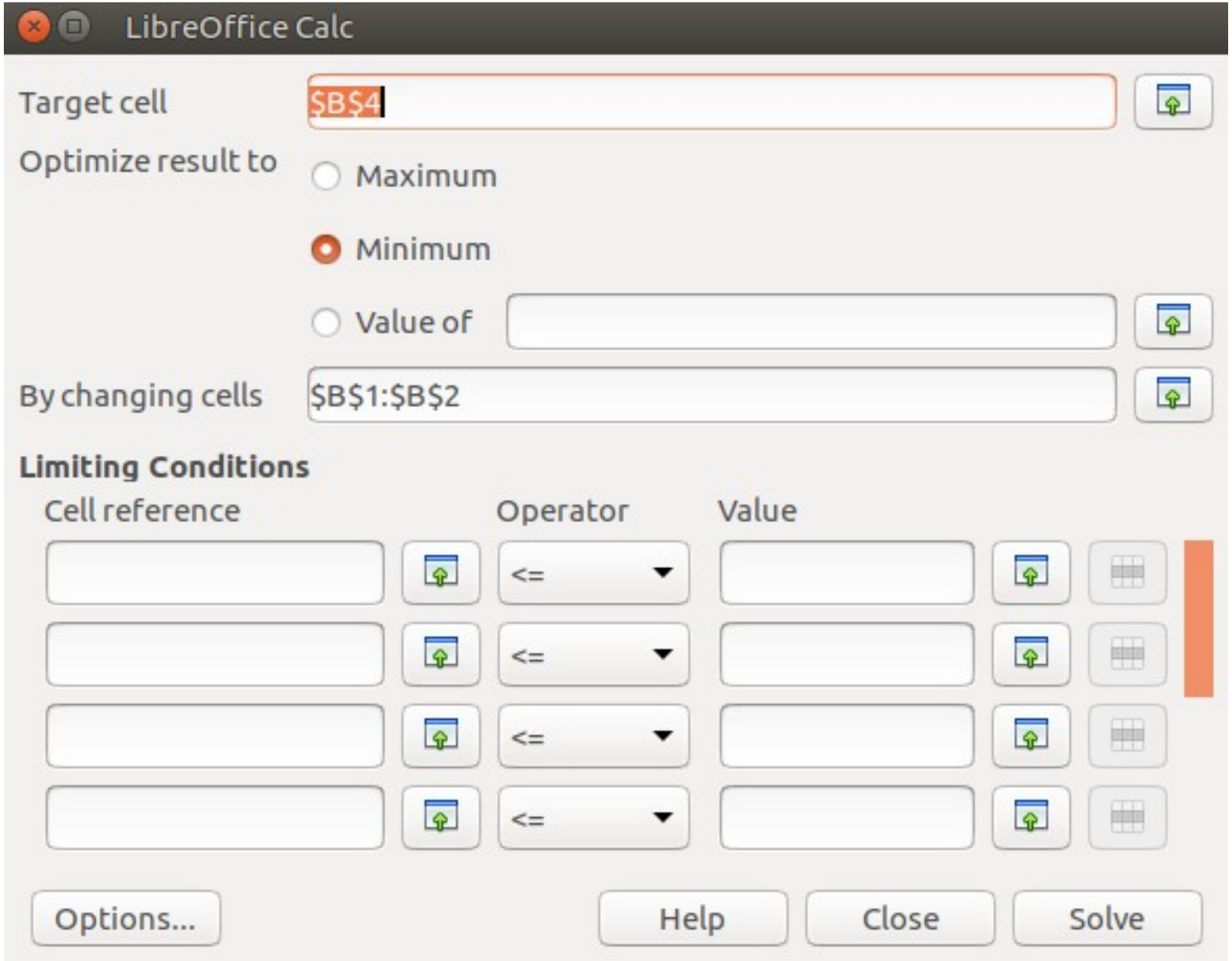
LibreOffice Calc: Solver

- Default linear Solver cannot be used to solve non-linear problem
- Need to download and install non-linear solver
- https://extensions.openoffice.org/fr/project/solver-nonlinear-programming-beta
- https://apps.ubuntu.com/cat/applications/precise/libreoffice-nlpsolver/
  - apt://libreoffice-nlpsolver
- Tool >> Extension Manager >> Add nlpsolver.oxt

Calc: Solver Extension

- First, I got error message during the installation
- Then, try to install by https://apps.ubuntu.com/cat/applications/precise/libreoffice-nlpsolver/
- Something is installed, but nothing pop-up yet. I retry the nlpsolver.oxt again, then, it works.




LibreOffice Calc
Target cell
$B$4
Optimize result to
Maximum
Minimum
Value of
By changing cells
$B$1:$B$2
Limiting Conditions
Cell reference
Operator
Value
<=
<=
<=
<=
Options...
Help
Close
Solve

	A	B	C	D	E
1	A	1		initial val	1
2	B	1			1
3	f(x) = A * (1 - exp(-B * x))				
4	SSE	2.655013146			
5					
6					
7		True val	True val		
8	x	fx Calc	fx Basic	predict	err^2
9	0	0	0	0	0
10	0.5	0.135951935	0.135951935	0.39346934	0.066315214
11	1	0.247			
12	1.5	0.338			
13	2	0.413			
14	2.5	0.474			
15	3	0.524			
16	3.5	0.565			
17	4	0.598			
18	4.5	0.626			
19	5	0.648			
20	5.5	0.666			
21	6	0.681			
22	6.5	0.694			
23	7	0.704			
24	7.5	0.712			
25	8	0.719			
26	8.5	0.724			
27	9	0.729			
28	9.5	0.733			
29	10	0.736			
30	10.5	0.738			
31	11	0.740			
32	11.5	0.742			
33	12	0.74			

LibreOffice Calc

Target cell: $B$4

Optimize result to: Maximum / Minimum / Value of

By changing cells: $B$1:$B$2

Limiting Conditions

Cell reference	Operator	Value
	<=	
	<=	
	<=	
	<=	

Options... Help Close Solve

LibreOffice Calc

Solver engine: DEPS Evolutionary Algorithm

Settings:

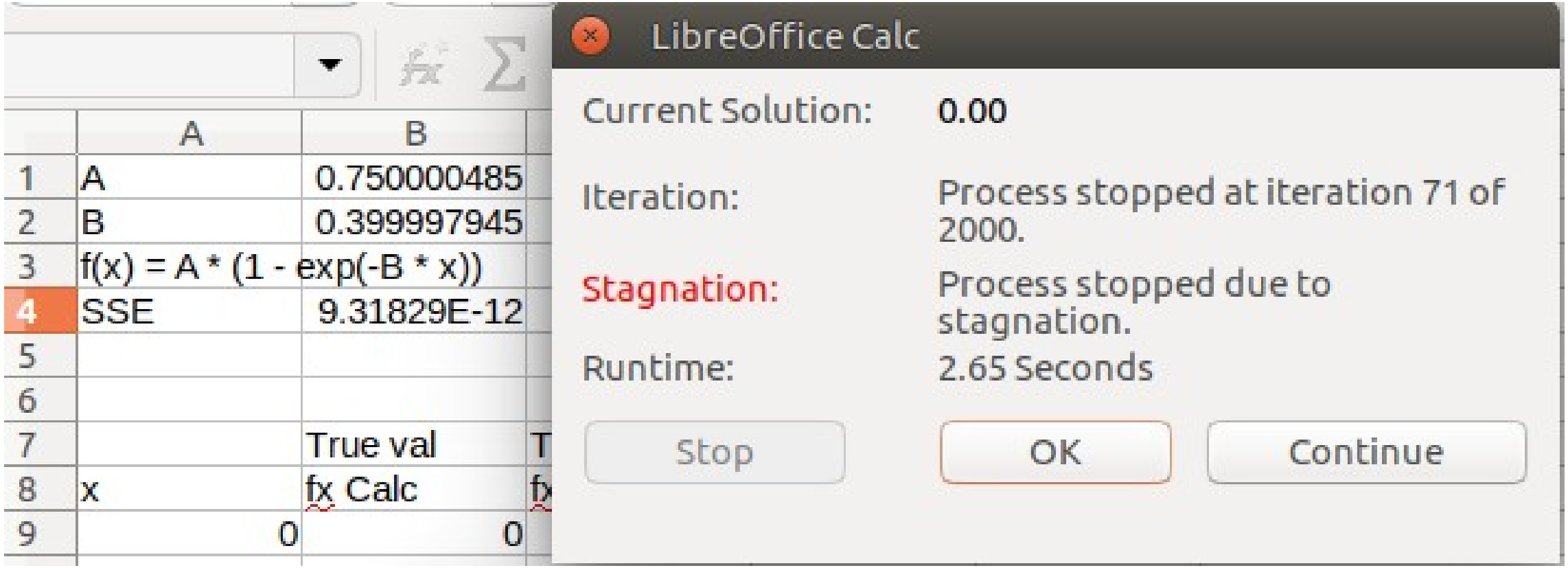
- Agent Switch Rate (DE Probability): **0.5**
- [x] Assume Non-Negative Variables
- DE: Crossover Probability (0-1): **0.9**
- DE: Scaling Factor (0-1.2): **0.5**
- Learning Cycles: **2000**
- PS: Cognitive Constant: **1.494**
- PS: Constriction Coefficient: **0.729**
- PS: Mutation Probability (0-0.005): **0**
- PS: Social Constant: **1.494**
- [x] Show enhanced solver status
- Size of Swam: **70**
- Stagnation Limit: **70**
- Stagnation Tolerance: **0.000000001**
- [ ] Use ACR Comparator (instead of BCH)
- [ ] Use Random starting point
- [x] Variable Bounds Guessing
- Variable Bounds Threshold (when guessing): **3**

Edit...

Help OK Cancel

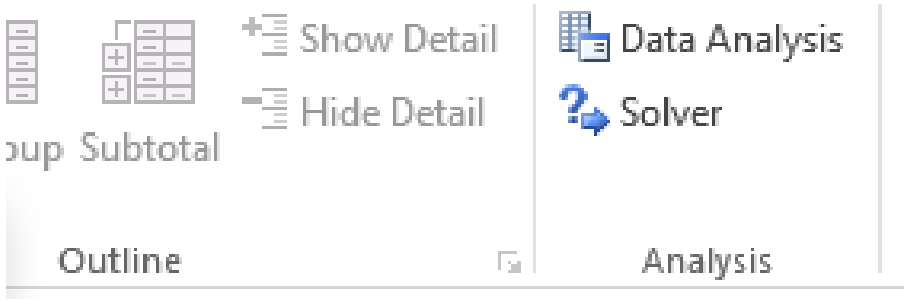
LibreOffice Calc Solver

- Check Sum Square Error (reach solution in 2.65 seconds)
- 9.3E-12 is very low already

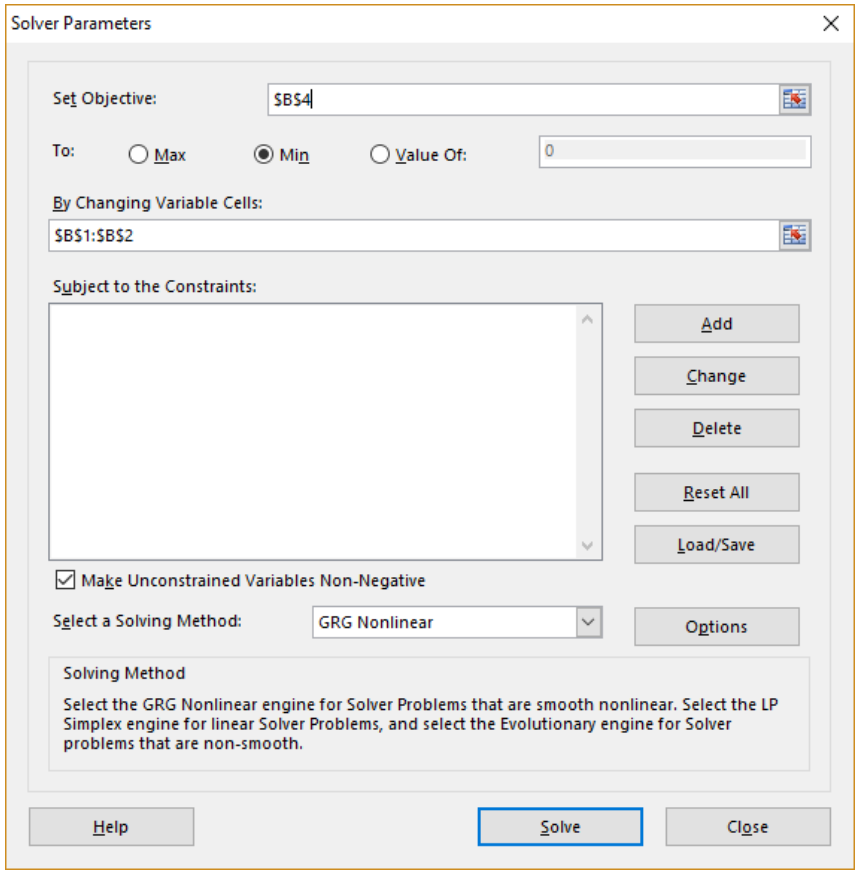


Excel Solver: Step-By-Step …

- Data >> Solver



- Set Objective: $B$4
- To: Min
- By Changing Variable Cells: $B$1:$B$2
- Select a Solving Method:
  - GRG NonLinear



Excel Solver: Step-By-Step …

- Solver help us to adjust the unknown constant in the equation until the sum-square-error becomes the minimum value.
- Then, we can do a curve fitting with non-linear equation

Problem:

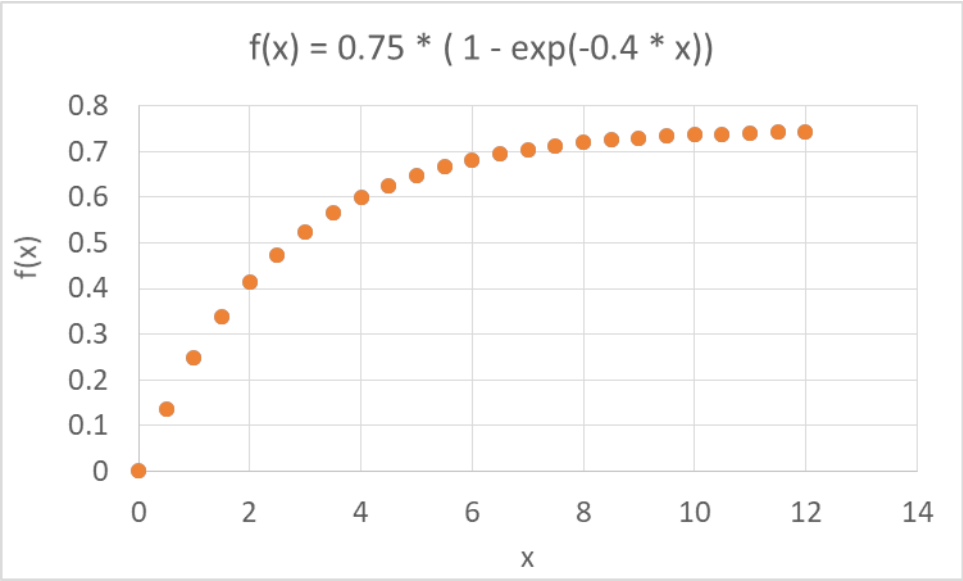
- What kind of non-linear equation that should be used for the curve fitting

One of the solution:

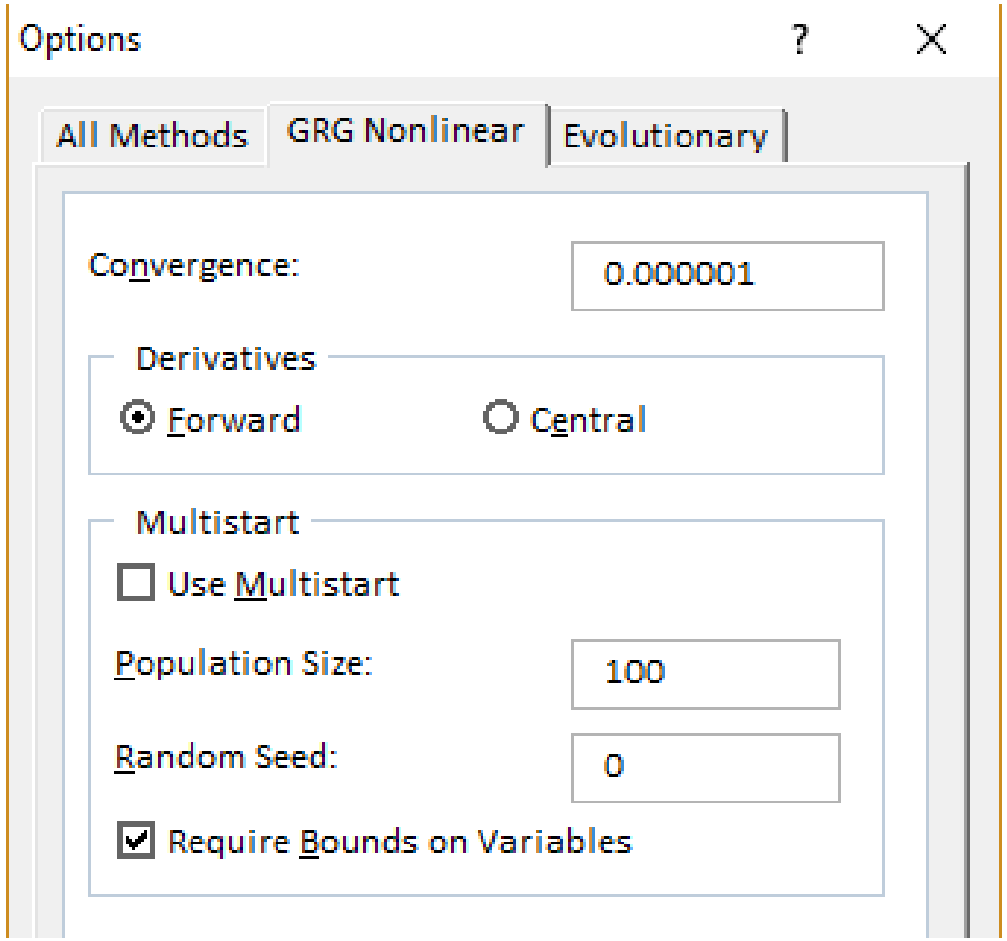
- Use LabFit: http://zeus.df.ufcg.edu.br/labfit/
- For Linux, you may try: goo.gl/ndN9eX

Excel Solver: Result

- You may decrease convergence criterion to get more accuracy

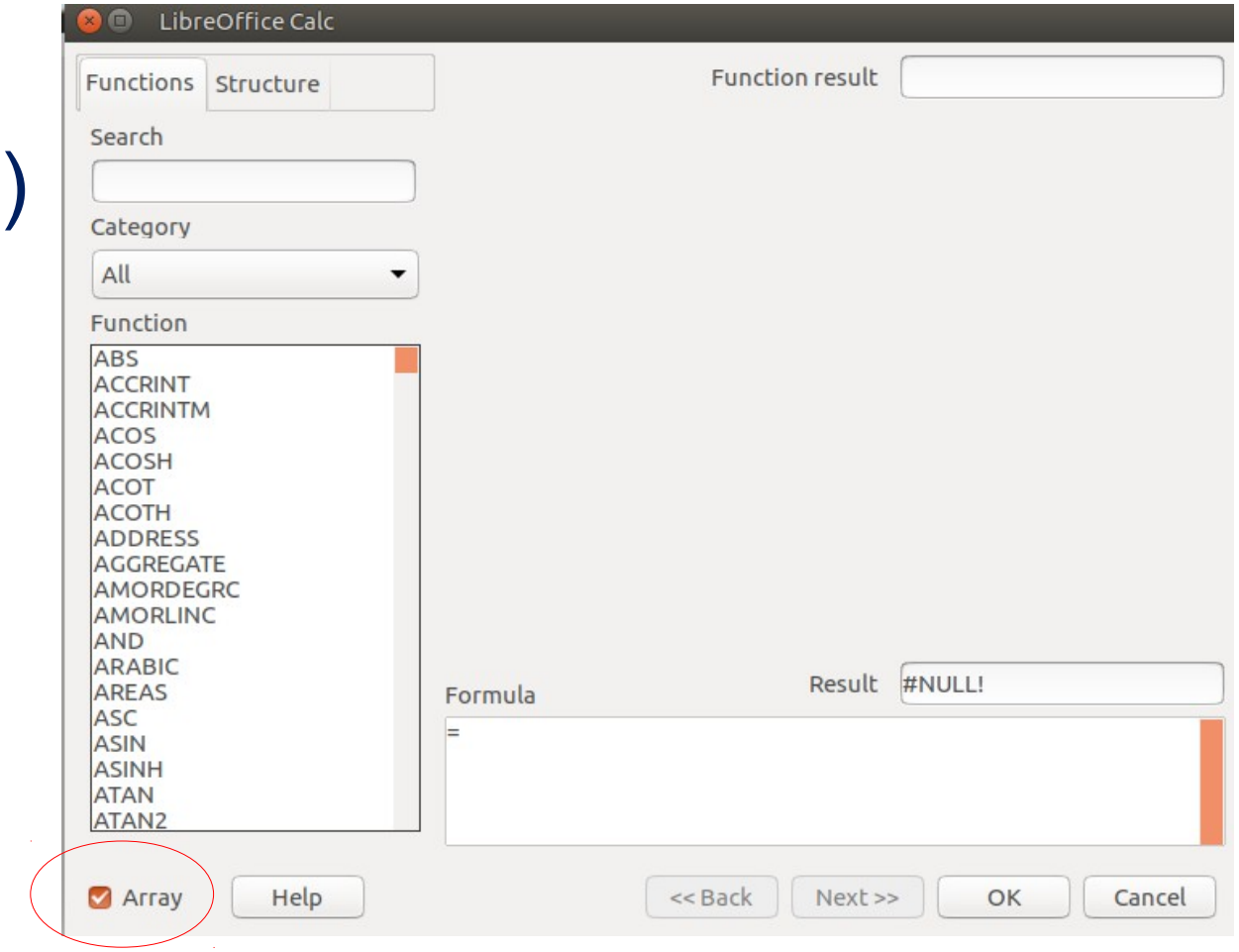


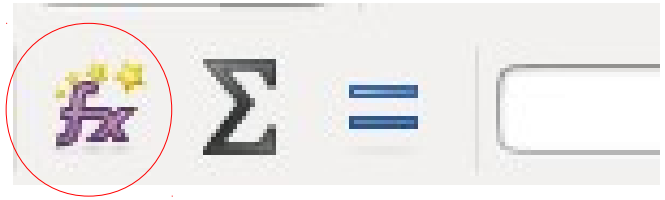
	A	B	C	D	E
1	A	0.75		initial val	1
2	B	0.400001			1
3	f(x) = A * (1 - exp(-B * x))				
4	SSE	9.87E-13			



LibreOffice Calc: Matrix Transpose: L01C04

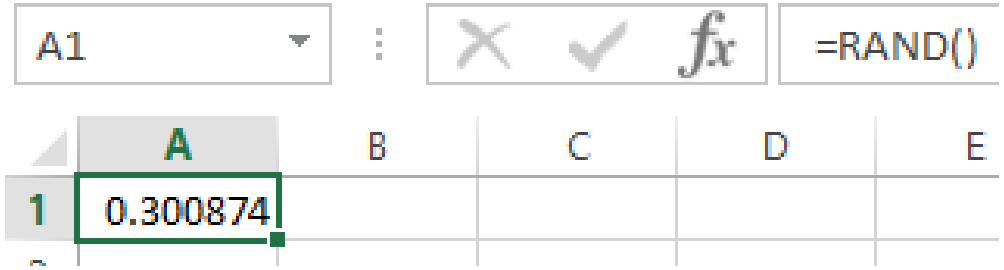
- Create 6 x 3 matrix with random number (use =RAND())
- Then copy to A1:C6. Copy again, but paste just value (so it does not change any more)
- Click on E1
- Type =Transpose(A1:C6)
- Press Shift + Ctrl + Enter
- Otherwise, select array option in function wizard

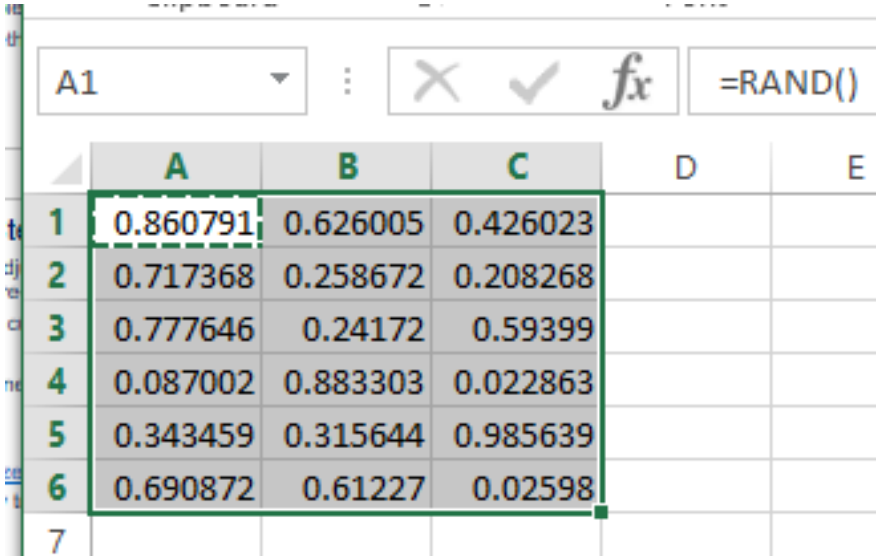




Excel: Matrix Transpose: L01C04

- Create 6 x 3 matrix with random number (use =RAND())
- Then copy to A1:C6. Copy again, but paste just value (so it does not change any more)
- Click on E1
- Type =Transpose(A1:C6)
- Press Enter, get #VALUE! (it is OK)
- Highlight E1:J3
- Press F2 + Ctrl + Shift + Enter





Liberation Sans 10

A1 =RAND()

	A	B	C	D	E	F
1	0.87605	0.20646	0.7382			
2	0.56246	0.55027	0.02203			
3	0.67671	0.34268	0.62399			
4	0.62456	0.24547	0.96233			
5	0.16059	0.14746	0.92841			
6	0.65456	0.15238	0.22781			
7						
8						
9	0.87605	0.56246	0.67671	0.62456	0.16059	0.65456
10	0.20646	0.55027	0.34268	0.24547	0.14746	0.15238
11	0.7382	0.02203	0.62399	0.96233	0.92841	0.22781

Excel: Matrix Transpose: L01C04: Result

- This converses M_{ij} to M_{ji}

	A	B	C	D	E	F	G	H	I	J
1	0.16388	0.14435	0.60426		0.16388	0.25844	0.77882	0.21467	0.06443	0.333
2	0.25844	0.27871	0.6876		0.14435	0.27871	0.68488	0.30909	0.61761	0.89094
3	0.77882	0.68488	0.64004		0.60426	0.6876	0.64004	0.75236	0.63451	0.66459
4	0.21467	0.30909	0.75236							
5	0.06443	0.61761	0.63451							
6	0.333	0.89094	0.66459							

E1 | {=TRANSPOSE(A1:C6)}

	A	B	C	D	E	F	G	H	I	J
1	0.1638780949	0.1443503997	0.6042586803		=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)
2	0.2584358862	0.2787102626	0.6876042718		=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)
3	0.7788224861	0.6848832517	0.6400440776		=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)	=TRANSPOSE(A1:C6)
4	0.2146671301	0.3090918536	0.7523601974							
5	0.0644343997	0.6176058108	0.6345139181							
6	0.3329988827	0.8909419990	0.6645913562							

Excel/Calc: Matrix-Matrix Operation: L01C04S2

- Excel: F2 + Ctrl + Shift + Enter
  - Just press enter without F2 + Ctrl + Shift + Enter will give just one output value or error message (do this first is OK)
  - Highlight enough output cells then press F2 + Ctrl + Shift + Enter while the active cell has the formula
- Calc: Shift + Ctrl + Enter

Matrix-Matrix Multiplication: use MMULT with

Matrix Inversion: use MINVERSE

Excel: Matrix-Matrix Operation…

- Normal View
- $MM^{-1} = I$

- Actual Value (Ctrl + `)

This part can be found in sheet 2 of L01C04.xlsm

	A	B	C	D
1	Initial matrix			
2	1	2	3	4
3	1	4	9	16
4	0	4.66666667	10	17.3333
5	0	2.33333333	3.33333333	4.33333
6				
7	MINVERSE			
8	0.25	0.75	-0.75	-7.1E-15
9	-16.25	16.25	-15.75	18
10	22.75	-22.75	21.75	-24
11	-8.75	8.75	-8.25	9
12				
13	MMULT			
14	1	0	0	0
15	0	1	2.8422E-14	0
16	0	0	1	0
17	-7.1054E-15	7.1054E-15	7.1054E-15	1

	A	B	C	D
1	Initial matrix			
2	1	2	3	4
3	=A2^2	=B2^2	=C2^2	=D2^2
4	0	=B3+B2/3	=C3+C2/3	=D3+D2/3
5	=A4*A2/A3	=B4*B2/B3	=C4*C2/C3	=D4*D2/D3
6				
7	MINVERSE			
8	=MINVERSE(A2:D5)	=MINVERSE(A2:D5)	=MINVERSE(A2:D5)	=MINVERSE(A2:D5)
9	=MINVERSE(A2:D5)	=MINVERSE(A2:D5)	=MINVERSE(A2:D5)	=MINVERSE(A2:D5)
10	=MINVERSE(A2:D5)	=MINVERSE(A2:D5)	=MINVERSE(A2:D5)	=MINVERSE(A2:D5)
11	=MINVERSE(A2:D5)	=MINVERSE(A2:D5)	=MINVERSE(A2:D5)	=MINVERSE(A2:D5)
12				
13	MMULT			
14	=MMULT(A2:D5,A8:D11)	=MMULT(A2:D5,A8:D11)	=MMULT(A2:D5,A8:D11)	=MMULT(A2:D5,A8:D11)
15	=MMULT(A2:D5,A8:D11)	=MMULT(A2:D5,A8:D11)	=MMULT(A2:D5,A8:D11)	=MMULT(A2:D5,A8:D11)
16	=MMULT(A2:D5,A8:D11)	=MMULT(A2:D5,A8:D11)	=MMULT(A2:D5,A8:D11)	=MMULT(A2:D5,A8:D11)
17	=MMULT(A2:D5,A8:D11)	=MMULT(A2:D5,A8:D11)	=MMULT(A2:D5,A8:D11)	=MMULT(A2:D5,A8:D11)

Programming with 2D Array

To complete this basic programming part, the learner should be able to

- Passing parameters to and from function
- Passing 2D array parameters
- Passing 2D array between VBA/OOB `Function` and `Worksheet`
- Understand how to use `For` Loop with 2D array

Let’s see more example / practice more with

- Matrix Transpose and Matrix Multiplication

VBA: Transpose

- Shape of matrix is given in B1 and B2
- B1 = number of row
- B2 = number of column
- The first (top left) value in matrix is given in B5
- Output the transpose such that it won’t overlap with the given matrix value with 2 cell whitespace gap.

VBA: Transpose: L01C05

- O1_a: subroutine to calculate transpose
- O1_b: same as O1_b, but shorter
- O1_c: show how to send and get array from a function
- O1_d: function used for O1_c
- O1_e: function to be called from the worksheet
  - Need to do F2 Ctrl Shift Enter

L01C05_O1_a: part(1/2)

```
Option Explicit
Sub L01C05_O1_a() 'full
    Dim ori_Mat() As Double
    Dim out_Mat() As Double
    Dim ori_N_row As Long
    Dim ori_N_col As Long
    Dim i As Long
    Dim j As Long

    ori_N_row = Cells(1, 2) 'number of row
    ori_N_col = Cells(2, 2) 'number of column
    ReDim ori_Mat(1 To ori_N_row, 1 To ori_N_col)
    ReDim out_Mat(1 To ori_N_col, 1 To ori_N_row)
```

L01C05_O1_a: part(2/2)

```
    For i = 1 To ori_N_row
        For j = 1 To ori_N_col
            'read original matrix
            ori_Mat(i, j) = Cells(4 + i, 1 + j)
            out_Mat(j, i) = ori_Mat(i, j)
        Next j
    Next i

    Cells(7 + ori_N_row, 1) = "Output"
    For i = 1 To ori_N_col
        For j = 1 To ori_N_row
            Cells(6 + ori_N_row + i, 1 + j) = out_Mat(i, j)
        Next j
    Next i
End Sub
```

```
Sub L01C05_01_b() 'quick, less memory usage
    Dim i As Long 'but less obvious
    Dim j As Long
    Dim Nrow As Long
    Dim Ncol As Long
    Nrow = Cells(1, 2)
    Ncol = Cells(2, 2)
    Cells(7 + Nrow, 1) = "Output"
    For i = 1 To Ncol
        For j = 1 To Nrow
            Cells(6 + Nrow + i, 1 + j) = _
                Cells(j + 4, i + 1)
        Next j
    Next i
End Sub
```

```
Sub L01C05_01_c() 'modular approach
    Dim ori_Mat() As Double
    Dim out_Mat() As Double
    Dim ori_N_row As Long
    Dim ori_N_col As Long
    Dim i As Long
    Dim j As Long
    ori_N_row = Cells(1, 2) 'number of row
    ori_N_col = Cells(2, 2) 'number of column
    ReDim ori_Mat(1 To ori_N_row, 1 To ori_N_col)
    ReDim out_Mat(1 To ori_N_col, 1 To ori_N_row)

    For i = 1 To ori_N_row
        For j = 1 To ori_N_col
            'read original matrix
            ori_Mat(i, j) = Cells(4 + i, 1 + j)
        Next j
    Next i
```

L01C05_O1_c: (continue)

- The order of the argument passing to other function must match to the parameter's order of the receiving function

```
    out_Mat = matTranspose(ori_Mat, ori_N_row, ori_N_col)
    Cells(7 + ori_N_row, 1) = "Output"
    For i = 1 To ori_N_col
        For j = 1 To ori_N_row
            Cells(6 + ori_N_row + i, 1 + j) = out_Mat(i, j)
        Next j
    Next i
End Sub
```

Function for L01C05_O1_c

```
'function take matrix and output matrix
'L01C05_O1_d (part of O1_c)
Function matTranspose(inMat As Variant _
        , rowLen As Long, colLen As Long) As Variant
    Dim i As Long
    Dim j As Long
    ReDim ansMat(1 To colLen, 1 To rowLen) As Double
    For i = 1 To rowLen
        For j = 1 To colLen
            ansMat(j, i) = inMat(i, j)
        Next j
    Next i
    matTranspose = ansMat
End Function
```

Convert VBA to OpenOffice Basic

L01C05_O1 a to d

- Just add Option VBASupport 1, then works properly
  - For a to c
  - Except O1_d
  - Need to Dim before ReDim when use in OOB

VBA in OOB

```
Function matTranspose(inMat As Variant _
        , rowLen As Long, colLen As Long) As Variant
    Dim i As Long
    Dim j As Long
    Dim ansMat() as Double
    ReDim ansMat(1 To colLen, 1 To rowLen) As Double
    For i = 1 To rowLen
        For j = 1 To colLen
            ansMat(j, i) = inMat(i, j)
        Next j
    Next i
    matTranspose = ansMat
End Function
```

Added line, otherwise, runtime error

Transpose function to be called from workbook

```
'L01C05_01_e
'Run from worksheet with F2 + Ctrl + Shift + Enter
Function myMTrans(inMat As Variant) As Variant
    Dim i As Long
    Dim j As Long
    Dim colLen As Long
    Dim rowLen As Long
    colLen = inMat.Columns.Count
    rowLen = inMat.Rows.Count
    ReDim ansMat(1 To colLen, 1 To rowLen) As Double
    For i = 1 To rowLen
        For j = 1 To colLen
            ansMat(j, i) = inMat(i, j)
        Next j
    Next i
    myMTrans = ansMat
End Function
```

L01C05_O1_e: Output

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Input				Direct function call														
2	1	2	0		1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
3	2	2	9		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
4	3	2	8		0	9	8	7	6	0	9	8	7	6	0	9	8	7	6
5	4	2	7																
6	5	2	6																
7	1	2	0																
8	2	2	9																
9	3	2	8																
10	4	2	7																
11	5	2	6																
12	1	2	0																
13	2	2	9																
14	3	2	8																
15	4	2	7																
16	5	2	6																

Direct Function Call: F2 + Ctrl Shift Enter

E2 | {=mymtrans(A2:C16)}

	A	B	C	D	E	F	G
1	Input				Direct function call		
2	1	2	0		=mymtrans(A2:C16)	=mymtran	=mym
3	2	2	9		=mymtrans(A2:C16)	=mymtran	=mym
4	3	2	8		=mymtrans(A2:C16)	=mymtran	=mym
5	4	2	7				
6	5	2	6				
7	1	2	0				
8	2	2	9				
9	3	2	8				
10	4	2	7				
11	5	2	6				
12	1	2	0				
13	2	2	9				
14	3	2	8				
15	4	2	7				
16	5	2	6				

myMTrans with Option VBASupport 1

- Press Shift + Ctrl + Enter (instead of just Enter)

RAND =mymtrans(A2:C16)

	A	B	C	D	E	F	G	H	I	J
1	Input				OO Basic					
2	1	2	0			=mymtrans(A2:C16)				
3	2	2	9							
4	3	2	8							
5	4	2	7							
6	5	2	6							
7	1	2	0							
8	2	2	9							
9	3	2	8							
10	4	2	7							
11	5	2	6							
12	1	2	0							
13	2	2	9							
14	3	2	8							
15	4	2	7							
16	5	2	6							

E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
OO Basic															
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	0	9	8	7	6	0	9	8	7	6	0	9	8	7	6

```
'Run from worksheet with Shift + Ctrl + Enter
Function myMTrans(inMat As Variant) As Variant
    Dim i As Long
    Dim j As Long
    Dim colLen As Long
    Dim rowLen As Long
    colLen = inMat.Columns.Count
    rowLen = inMat.Rows.Count
    Dim ansMat() As Double
    ReDim ansMat(1 To colLen, 1 To rowLen) As Double
    For i = 1 To rowLen
        For j = 1 To colLen
            ansMat(j, i) = inMat(i, j)
        Next j
    Next i
    myMTrans = ansMat
End Function
```

OOBasic Version

First line, Need
`Option VBASupport 1`

Added line, otherwise, runtime error

More Programming Practice

- To fully appreciate and understand how to do VBA programming, perform matrix multiplication with and without using MMULT in VBA
- O2_a: matrix multiplication subroutine
  - Show how to use range
- O2_b: use Excel `MMULT` function
  - Show how to use `WorksheetFunction`
  - Use `UBound` and `LBound` to get array size (no need to send size together with matrix anymore)

myMMult part 1/2

```
'L01C05_O2_a
Sub myMMult()
    Dim mat1() As Double
    Dim mat2() As Variant
    Dim ansMat(1 To 5, 1 To 3)
    Dim k As Long
    Dim sum As Double
    Dim i As Long
    Dim j As Long
    ReDim mat1(1 To 5, 1 To 2) As Double
    For i = 1 To 5
        For j = 1 To 2
            mat1(i, j) = Cells(3 + i, j)
        Next j
    Next i
    mat2 = Range("D4:F5")
```

```
    For i = 1 To 5
        For j = 1 To 3
            sum = 0
            For k = 1 To 2
                sum = sum + mat1(i, k) * mat2(k, j)
            Next k
            ansMat(i, j) = sum
        Next j
    Next i
    Cells(13, 1) = "fn_output"
    For i = 1 To 5
        For j = 1 To 3
            Cells(i + 12, j + 1) = ansMat(i, j)
        Next j
    Next i
End Sub
```

myMMult part 2/2

	A	B	C	D
19				
20	Excel: ans	5	6.5	8
21		10	13	16
22		15	19.5	24
23		20	26	32
24		25	32.5	40

	A	B	C
19			
20	Excel: ans	=MMULT(A4:B8,D4:F5)	=MMULT(A4:
21		=MMULT(A4:B8,D4:F5)	=MMULT(A4:
22		=MMULT(A4:B8,D4:F5)	=MMULT(A4:
23		=MMULT(A4:B8,D4:F5)	=MMULT(A4:
24		=MMULT(A4:B8,D4:F5)	=MMULT(A4:

	A	B	C	D	E	F
1						
2	Mat(5 x 2) * Mat(2 x 3) = Mat(5 x 3)					
3						
4	1	0.5		4	5	6
5	2	1		2	3	4
6	3	1.5				
7	4	2				
8	5	2.5				
9						
10						
11	A_ij * B_jk = A_ik					
12						
13	fn_output	5	6.5	8		
14		10	13	16		
15		15	19.5	24		
16		20	26	32		
17		25	32.5	40		

L01C05_O2_b: Part(1/2)

```
'L01C05_O2_b
Sub myMMult_autoFN()
    Dim mat1() As Double
    Dim mat2() As Double
    Dim ansMat As Variant
    Dim i As Long
    Dim j As Long
    ReDim mat1(1 To 5, 1 To 2) As Double
    ReDim mat2(1 To 2, 1 To 3) As Double
    For i = 1 To 5
        For j = 1 To 2
            mat1(i, j) = Cells(3 + i, j)
        Next j
    Next i
    For i = 1 To 2
        For j = 1 To 3
            mat2(i, j) = Cells(3 + i, 3 + j)
        Next j
    Next i
```

L01C05_O2_b: Part(2/2)

```
    ansMat = WorksheetFunction.MMult(mat1, mat2)
    Cells(26, 1) = "use WorksheetFunction.MMult"

    Dim ansNRow As Long
    Dim ansNCol As Long
    ansNRow = UBound(ansMat, 1) - LBound(ansMat, 1) + 1
    ansNCol = UBound(ansMat, 2) - LBound(ansMat, 2) + 1
    For i = 1 To ansNRow
        For j = 1 To ansNCol
            Cells(26 + i, 1 + j) = ansMat(i, j)
        Next j
    Next i
End Sub
```

myMMult: OOBasic (without `VBASupport 1`)

New Tricks

- Use current active sheet
- Use no need to pass array shape (use UBound for shape)
- Output at the calling location
- Works the same way as MMULT

By Learning how to pass array to/from worksheet, you will be able to write a lot more complicated algorithms

myMMult: OOB Version

```
REM  *****  BASIC  *****
Option Explicit
'run by typing
'=MYMMULT_OOB(A4:B8,D4:F5)
Function myMMult_OOB(Mat1 As Variant, Mat2 As Variant) _
        As Variant
    Dim ansMat(1 to UBound(Mat1,1), _
        1 to UBound(Mat2,2)) as Double
    Dim sum as double
    Dim i as long
    Dim j as long
    Dim k as long
    Dim Sheet as Object
    Dim Cell as Object
```

```
    Sheet = thiscomponent.getcurrentcontroller. _
        activesheet
    Cell = Sheet.getCellByPosition(0,0)
    Cell.string = "test"
    'just to show another way to print
    For i = 1 to UBound(Mat1,1)
        For j = 1 to UBound(Mat2,2)
            sum = 0
            For k = 1 to UBound(Mat1,2)
                sum = sum + Mat1(i,k) * Mat2(k,j)
            Next k
            ansMat(i,j) = sum
        Next j
    Next i
    myMMult_OOB = ansMat
End Function
```

Output

A11:C15 {=MYMMULT_OOB(A4:B8,D4:F5)}

	A	B	C	D	E	F
1	test					
2	Mat(5 x 2) * Mat(2 x 3) = Mat(5 x 3)					
3						
4	1	0.5		4	5	6
5	2	1		2	3	4
6	3	1.5				
7	4	2				
8	5	2.5				
9						
10	Ans from myMMult					
11	5	6.5	8			
12	10	13	16			
13	15	19.5	24			
14	20	26	32			
15	25	32.5	40			
16						
17	Ans from Calc Mmult					
18	5	6.5	8			
19	10	13	16			
20	15	19.5	24			
21	20	26	32			
22	25	32.5	40			

Output & Expected Homework

- From myMTrans, a learner can write a function that take 3 arguments from workbook, and output the multiplication of those 3, without using any automatic function (use just for and if).
- This function should require F2+CSE / SCE to run

	A	B	C	D
19				
20	Excel	5	6.5	8
21		10	13	16
22		15	19.5	24
23		20	26	32
24		25	32.5	40
25				
26	use WorksheetFunction.MMult			
27		5	6.5	8
28		10	13	16
29		15	19.5	24
30		20	26	32
31		25	32.5	40

Expected Homework

- Create function taking value from worksheet and calculate (matrix ^ 4)^T
- Use solver to the parameter that make the
  - (Mat^4)^T = a certain 3x3 matrix
  - Will be asked to use solver together with VBA/OOo Function

Application with VBA/OOBasic

- Help to debug/develop program in Python / C++
- Help to directly show a complicated algorithm step-by-step via worksheet and user define function
- With just we learn about Worksheet + VBA/OOBasic, we can show the working principle in a step-by-step fashion for
  - Newton method, Gaussian elimination, RK4, RKF45, Trapezoidal/Simpson integration, Gradient Descent optimization, Levenberg Marquardt optimization, and many more numerical methods

Tool/Program/OS used for making these slides

- LibreOffice Calc/Impress Version 5.3.1.2
- Ubuntu 17.04 (64-bit, Intel Core i7, ram 8 GB)
- Windows 10 (Excel 2013)
- KolourPaint Version 16.12.3 (crop / copy-paste /edit picture)
- Take Screenshot Version 1.18.1-0ubuntu1