Financial Data Lab

E. Sofia Morote

Course Summary

In this course, Students access and manipulate a variety of financial market data.

The labs are in class sessions supervised by an instructor. Each lab session is devoted to:

One exercise involving the use and manipulation of data

Examples include:

Retrieving information on Treasury securities

Building a real-time options calculator in Excel

Retrieving and analyzing historical stock price, and comparing the cash and futures basis in real time.

Financial Data Lab

Assignment 1

Sofia Morote

Activities

1. **Obtain the current bids and asks for the 20 stocks that make up the major market index.**

2. **For each stock, find out what information is available on the company, whether options are traded on the stocks, and any dividend information.**

3. **Using the PTW and Excel, link the bid, ask and the last traded price into spreadsheet so they update automatically**

Document to turn in:

Report bids and ask for the 20 Stocks (listed on pg. 3).

Introduction (Con't)

SYSTEM DESCRIPTION

Personal Trading Workstation (PTW)

PTW includes up to four screens controlled by one keyboard and/or mouse. It can display and manipulate information from many different sources, which may be video or digital transmissions. The following applications are available with PTW:

Reuters IDN Data display

Data Dictionary, which provides linking of Triarch data into applications such as Excel via Dynamic Data Exchange (DDE)

Page Display for display of page-based servers available on Triarch 2000

Terminal Emulation for connection to in-house servers

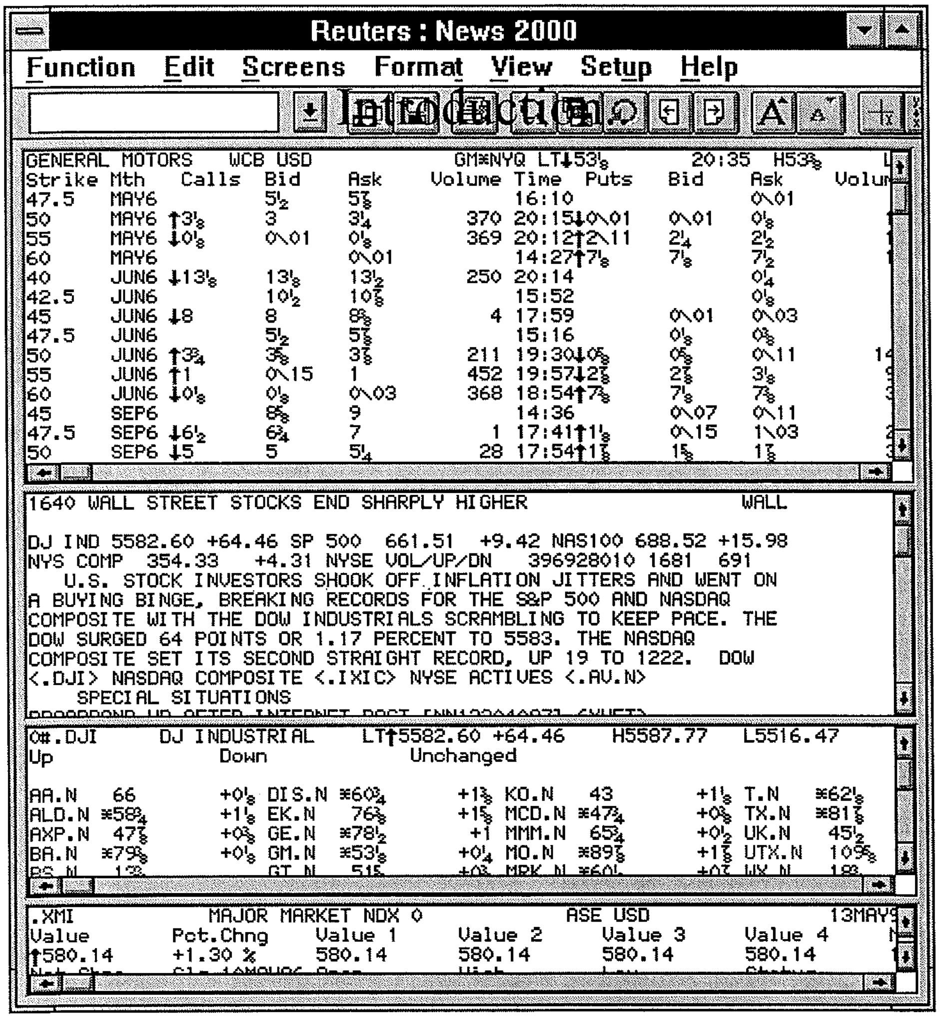
Workstation composite paging with the Montage application

24 hour News application with search and browse

Tic Graphs including Limit and Tolerance Minding on all Triarch sources

Study Graphics for graphical analysis of data

Introduction



A Reuter Instrument Code (RIC) is a unique code assigned to every company/instrument on the Reuters Network and used in conjunction with qualifiers to retrieve specific data from the Reuters network.

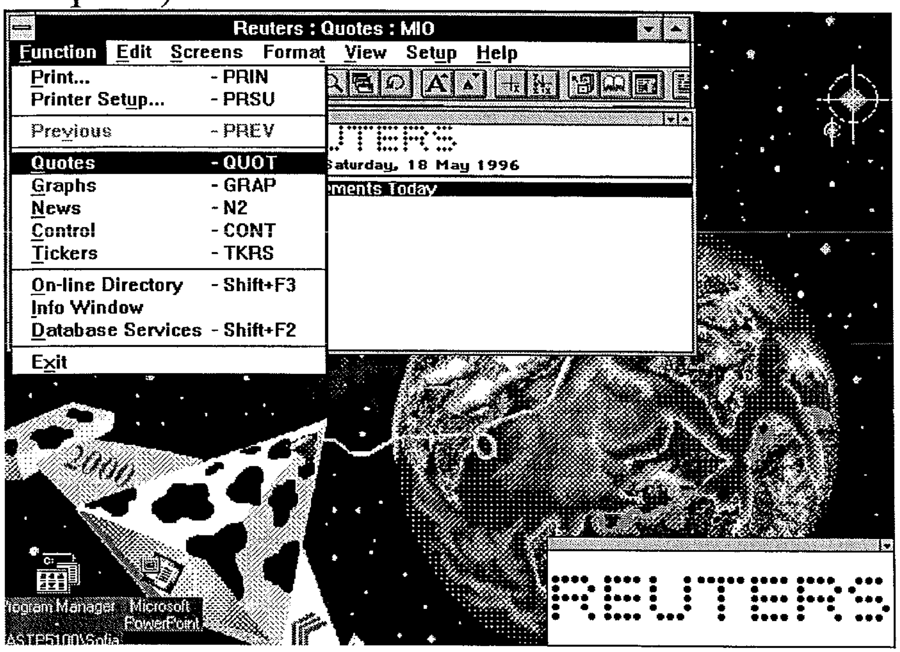
- Suggested RIC codes for the 20 MMI stocks:

AXP.N	T.N	CHV.N	KO.N	DIS.N
DOW.N	DD.N	EK.N	XON.N	GE.N
GM.N	IBM.N	IP.N	JNJ.N	MCD.N
MRK.N	MMM.N	MO.N	PG.N	S.N

Activity 1:The purpose of this activity is to obtain the current bids and asks for the 20 stocks that make up the major market index.

To accomplish this goal, first set up a Quote List

1. Double click on the icon PTW to enter the PTW program
2. Double-click on "Reuter Trade Workstation" to access the following screen (the background may vary depending on the computer)



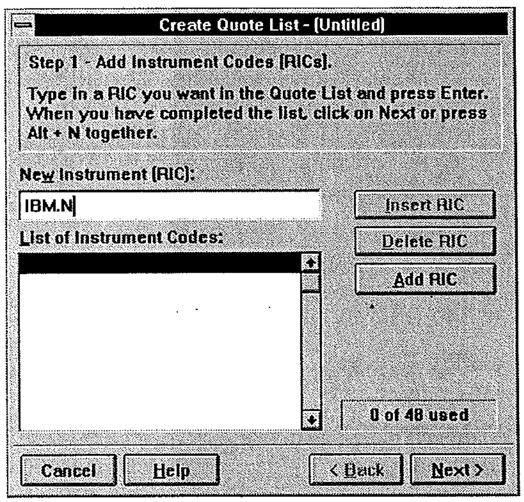
3.Select FUNCTION from the menu bar and select QUOTES from the pull-down menu.

4. Select SETUP from the menu bar and select QUOTE LIST from the pull-down menu then select NEW.

A "Create Quote List" dialogue box appears.

Activity 1 (con't)

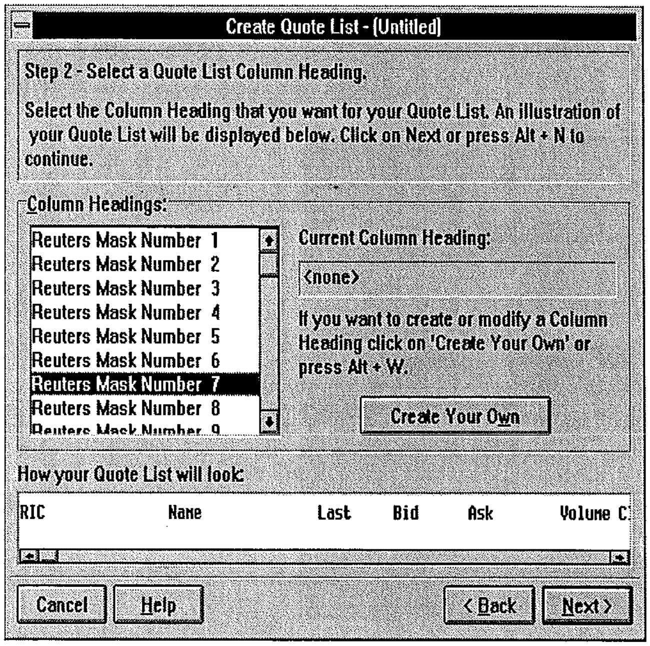
5. Type the first entry from the RIC Codes List (on pg. 3) into the box marked "New Instrument (RIC):". Press ENTER after this entry.



6. Enter each item on the RIC codes list, pressing ENTER between each item. (if you want to change an entry, scroll down the list & select the item you want to change or modify then make the modification).

When you have completed the list, click on "Next" (in the lower right hand corner) A dialogue box appears:

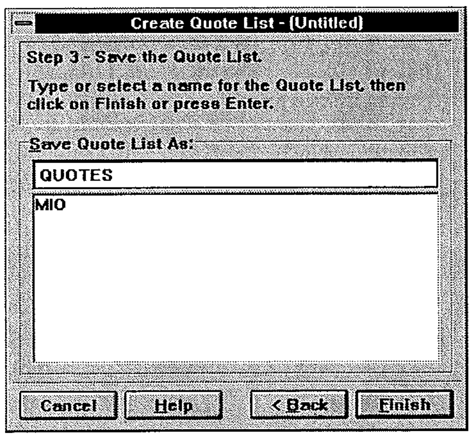
Activity 1 (con't)



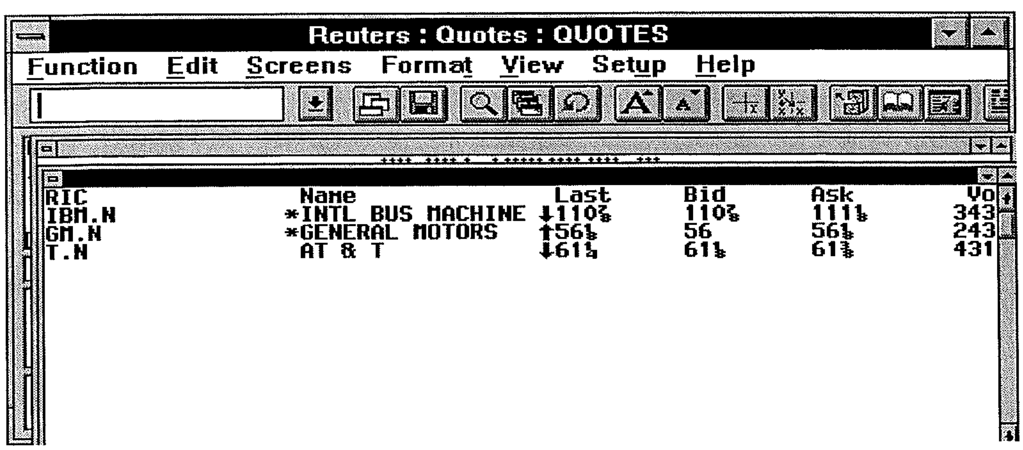
7. From the "Create Quote List" dialogue box (Step 2) Select a column heading (The columns will be labeled as you choose).

Activity 1 (con't)

8. Name the quoted list by typing a name into the "Save Quoted List As:" box
9. Hit ENTER to save your quoted list



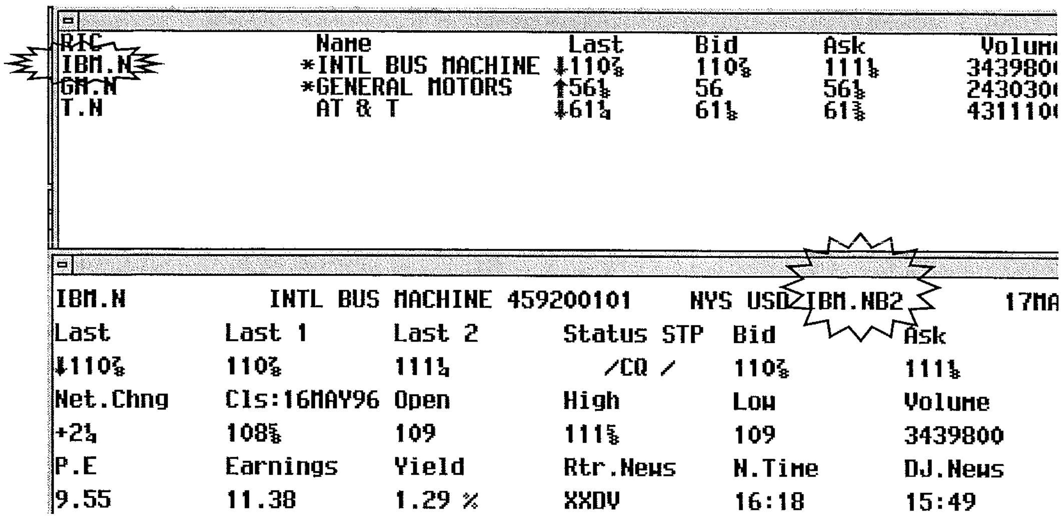
The quoted list will appear in the active window.



Activity 2: The purpose of this activity is to find out what information is available on the company, whether options are traded on the stocks, and any dividend information

To accomplish this goal, get information about each company.

1. For get the information on the company , you can double click on the RIC code (the company you want to learn about). An active window of full quoted will appear

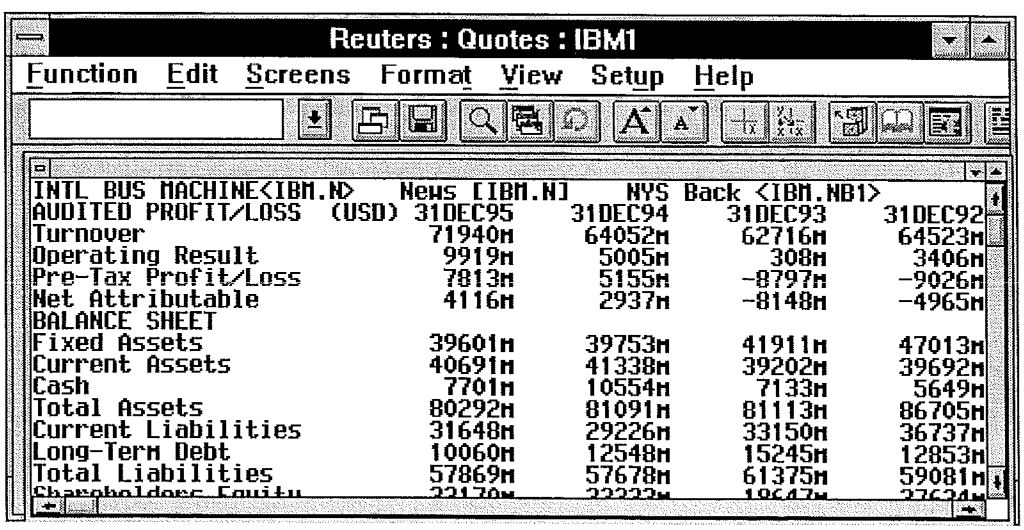


RIC	Name	Last	Bid	Ask	Volum
IBM.N	*INTL BUS MACHINE	↓110⅞	110⅞	111⅛	343980
GM.N	*GENERAL MOTORS	↑56⅛	56	56⅛	243030
T.N	AT & T	↓61¼	61⅛	61⅜	431110

IBM.N INTL BUS MACHINE 459200101 NYS USD IBM.NB2 17MA

Last	Last 1	Last 2	Status STP	Bid	Ask
↓110⅞	110⅞	111¼	/CQ /	110⅞	111⅛
Net.Chng	Cls:16MAY96	Open	High	Low	Volume
+2¼	108⅝	109	111⅝	109	3439800
P.E	Earnings	Yield	Rtr.News	N.Time	DJ.News
9.55	11.38	1.29 %	XXDV	16:18	15:49

2. Double click on the code of the code that follows "NYS USD" (in this case, IBM.NB2)

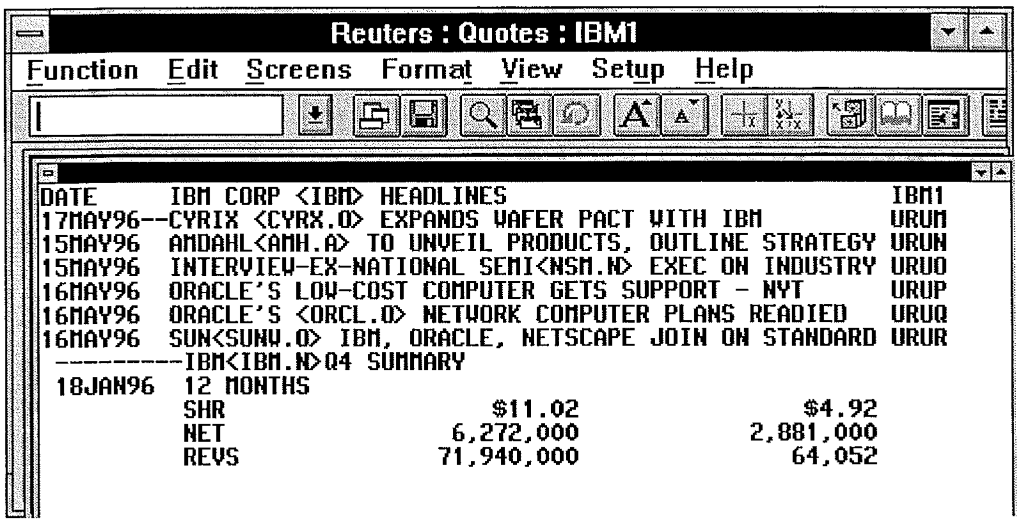


INTL BUS MACHINE<IBM.N> News [IBM.N] NYS Back <IBM.NB1>

AUDITED PROFIT/LOSS (USD)	31DEC95	31DEC94	31DEC93	31DEC92
Turnover	71940m	64052m	62716m	64523m
Operating Result	9919m	5005m	308m	3406m
Pre-Tax Profit/Loss	7813m	5155m	-8797m	-9026m
Net Attributable	4116m	2937m	-8148m	-4965m
BALANCE SHEET				
Fixed Assets	39601m	39753m	41911m	47013m
Current Assets	40691m	41338m	39202m	39692m
Cash	7701m	10554m	7133m	5649m
Total Assets	80292m	81091m	81113m	86705m
Current Liabilities	31648m	29226m	33150m	36737m
Long-Term Debt	10060m	12548m	15245m	12853m
Total Liabilities	57869m	57678m	61375m	59081m

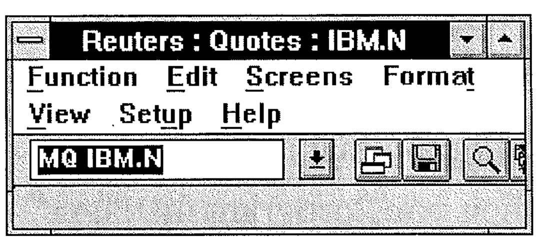
Activity 2 (con't)

3. Return to the full quotes window
4. Double click on "Headlines" to bring up information about recent headlines.



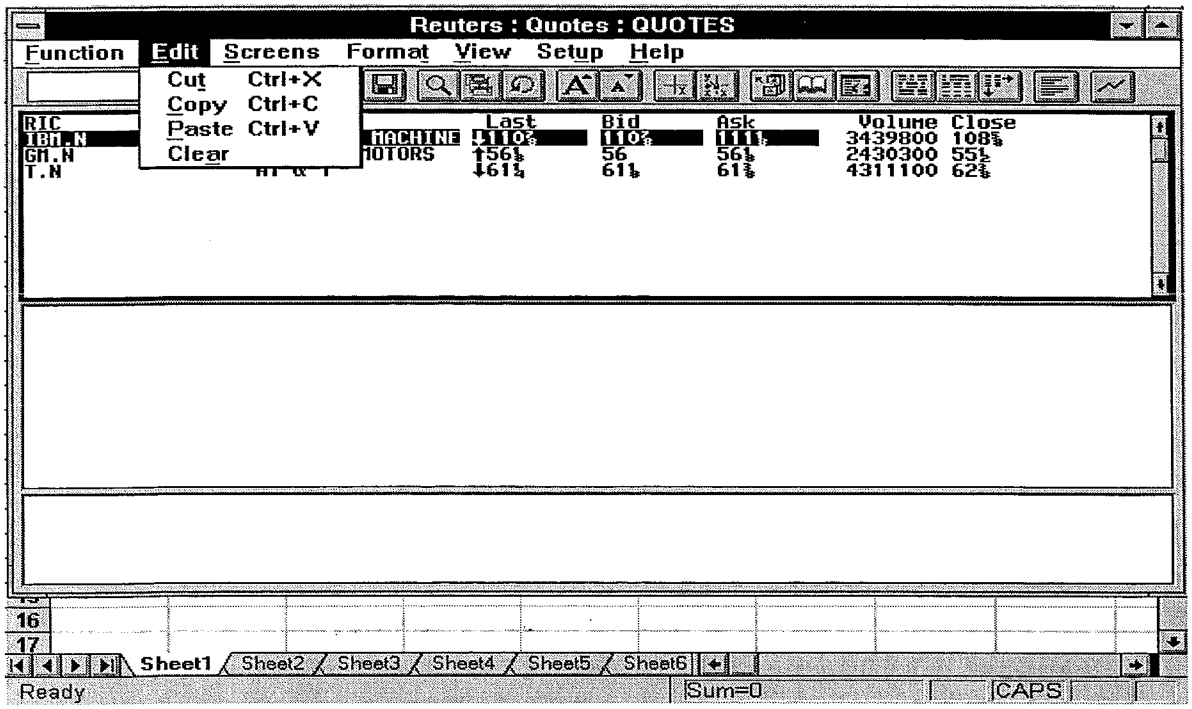
Suggestions:

- If you want to display the Mini Quote window, Type: MQ RIC in the command box, and press F5 or Return key (the command box is in upper left hand corner directly under the word "Function" in the menu)



Activity 3: Using the PTW and Excel, link the bid, ask and the last traded price into spreadsheet so they update automatically

1. Return to the active window (RIC, BID, ASK) that you built in Activity 1. Highlight any RIC code

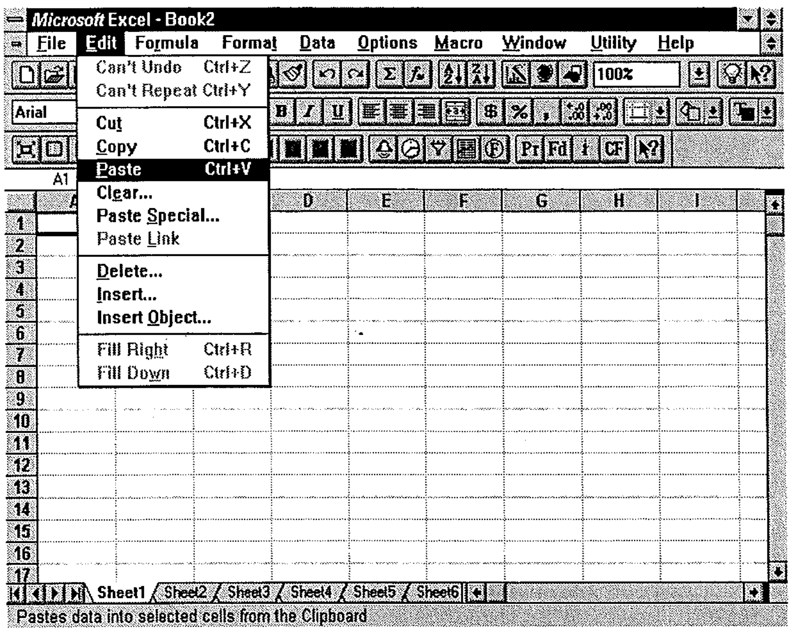


2. Select EDIT from the menu and select COPY from the pull-down menu (this copies the RIC code information)

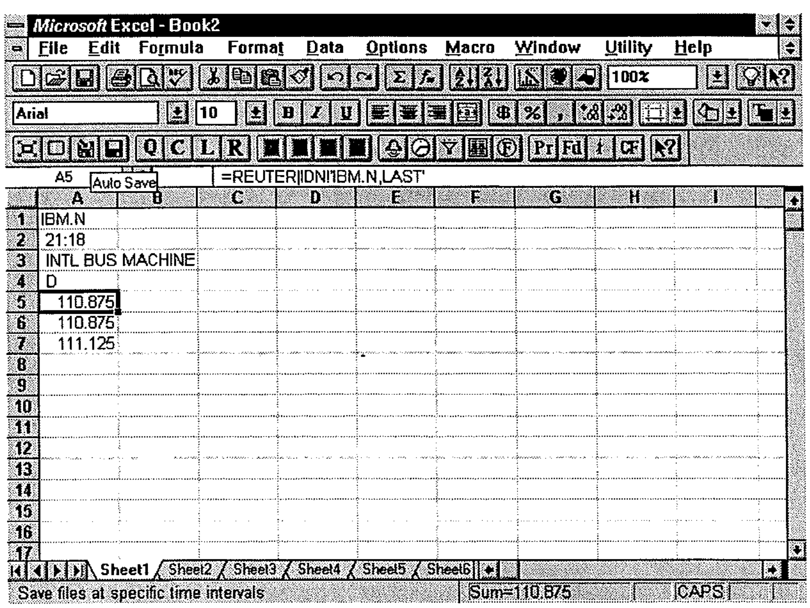
3. Press and hold "Alt" from the keyboard and the same time use "tab" to select "program manager" from the windows options. Release "Alt" when you "program manager" appears. Double-click on the Excel icon from the PTW screen.

Activity 3 (Con't)

4 .In EXCEL, Select EDIT from the menu bar and select PASTE from the pull-down menu



A screen with like the following appears, showing that a link has been created



Activity 3 (Con't)

- <u>Suggestion:</u>

A short way to do the link the bid, ask and last price is:

Open EXCEL

Write the formulas:

=REUTER|IDN!'RIC,BID'

=REUTER|IDN!'RIC,ASK'

=REUTER|IDN!'RIC,LAST'

Once these formulas have been entered, the lists update automatically.

Financial Data Lab

Assignment 2

Sofia Morote

Activities

1. Using the Reuterhist program available on the HP's, obtain weekly stock price data on the 20 MMI stocks and the S&P500 index. Get as much historical data as is available.

2. Translate the data using FTP or FILE EXPRESS

3. Import the Data into Excel spreadsheet

4. Plot the prices as a function of time

5. Calculate the average return on each stock and the standard deviation of returns, and

   Then compare these to the average return and standard deviation of the S&P500.

Assignment 2-Document to turn in

Report the average return on each stock , the standard deviation of the returns.

Compare these to the average return and standard deviation of the S&P500.

Based on the return and standard deviation of the returns, choose 5 stocks.

Plot the prices of the 5 stocks as a function of time: Do not use a full page for each graph plot at least 4 graphs per page.

Activity 1: Using the Reuterhist program available on the HP's, obtain weekly stock price data on the 20 MMI stocks and the S&P500 index. Get as much historical data as is available.

- Required RIC codes for the 20 MMI stocks:

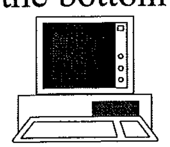
AXP.N	T.N	CHV.N	KO.N	DIS.N
DOW.N	DD.N	EK.N	XON.N	GE.N
GM.N	IBM.N	IP.N	JNJ.N	MCD.N
MRK.N	MMM.N	MO.N	PG.N	S.N

- Code for the S&P500 composite stock price

  .SPX

1. login as atw (using your login) to the HP, enter your password and hit "Enter"

   On the bottom line of the screen, locate a small icon like this:



(Note that the actual icon is much smaller)

2. Click on this icon

One of the following prompts will be returned

%

#[atw]\atw:

If you received a "#[atw]\atw" prompt, then enter:

```
#[atw]\atw: reuterhist <outputfile> d/w/m RIC [RICS]
```

If you received a "%" prompt, then enter :

```
% reuterhist <outputfile> d/w/m RIC [RICS]
```

Activity 1 (Con't)

Where:

"outputfile": is the name of the file to write

d/w/m : is your selection of daily, weekly, or monthly data

RIC: (Reuters Instrument Codes), represents the specific RIC code(s) you want to get information about (in both cases, you can enter any number of RIC codes, as long as you leave one blank space between each code.

Once the program is done fetching the data it will report ..".all done"

Activity 2: Translate the data using FTP or FILE EXPRESS

For this activity use the **PC**

Method 1: using FILE EXPRESS:

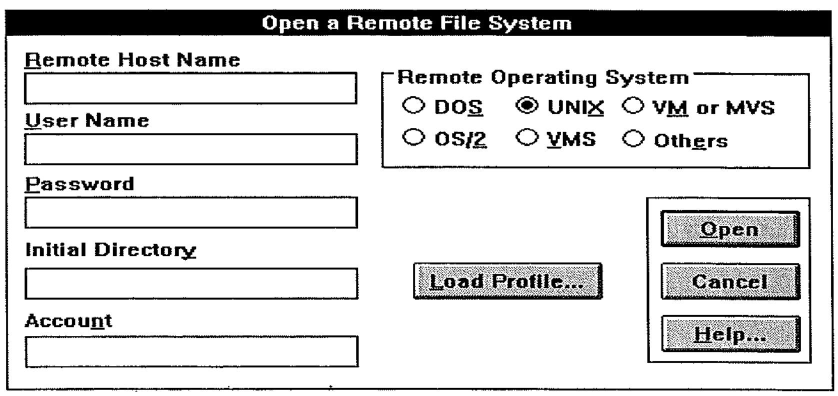
1. From the Program Manager, Double-click on LAN Workplace
2. From the LAN Workplace, when double -click on File Express. A screen appears, fill in the required information as follows:

Remote host name: name of HP

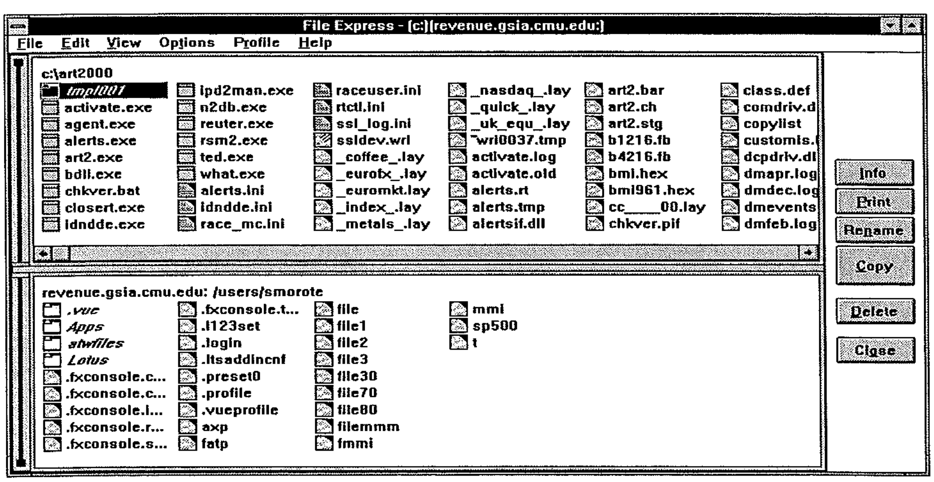
User name: your atw (your login)

Password: your password (Do not fill in any other fields)

Press "Enter"



A double screen appears (as shown below)

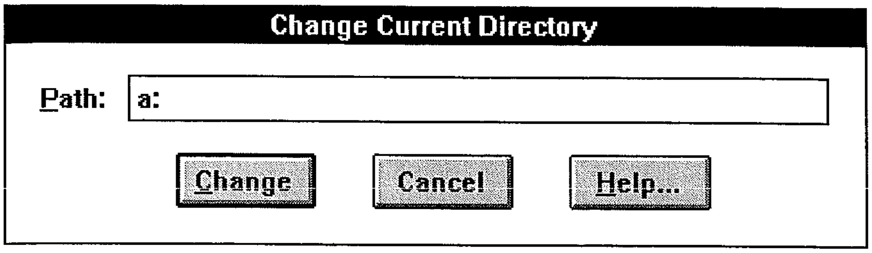


Activity 2 (Con't)

3. In the top screen, click on "c:\" (This is located in the upper left corner of one the screens)

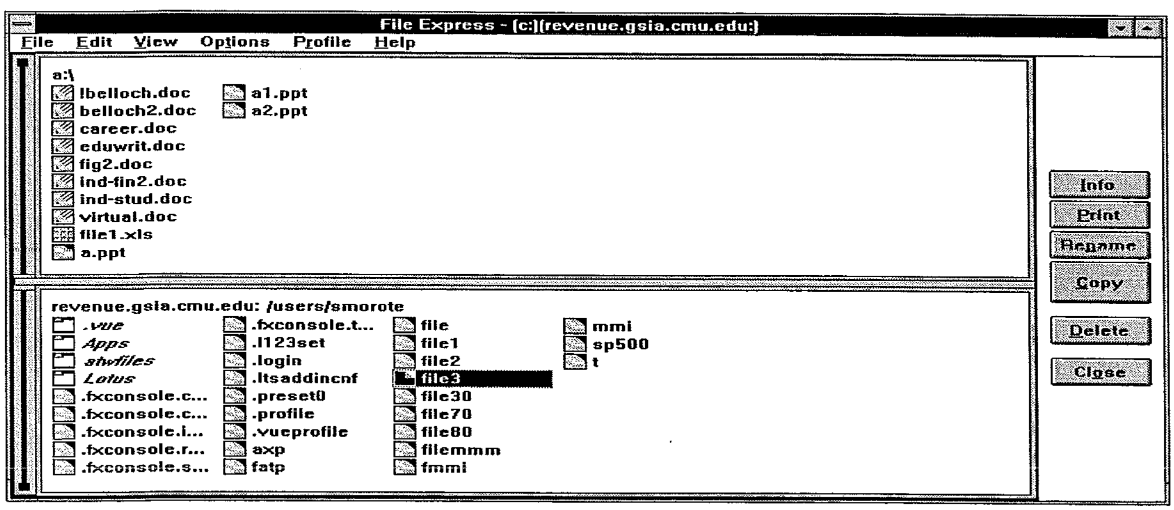
Once you have clicked on "c:\", the following dialogue box will appear.

4. Type in "a:" after "path" press "Enter" (this will allow you to copy the information to your diskette)



Once you have hit "Enter" a list of files will appear.

5. Select your output file (e.g. File 3) Double-click on "Copy" from the buttons on the right side of the screen.

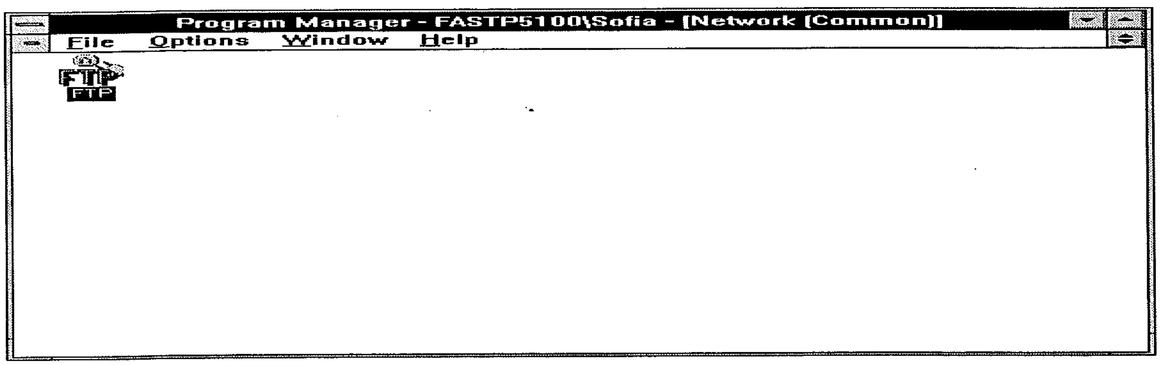


Activity 2 (Con't)

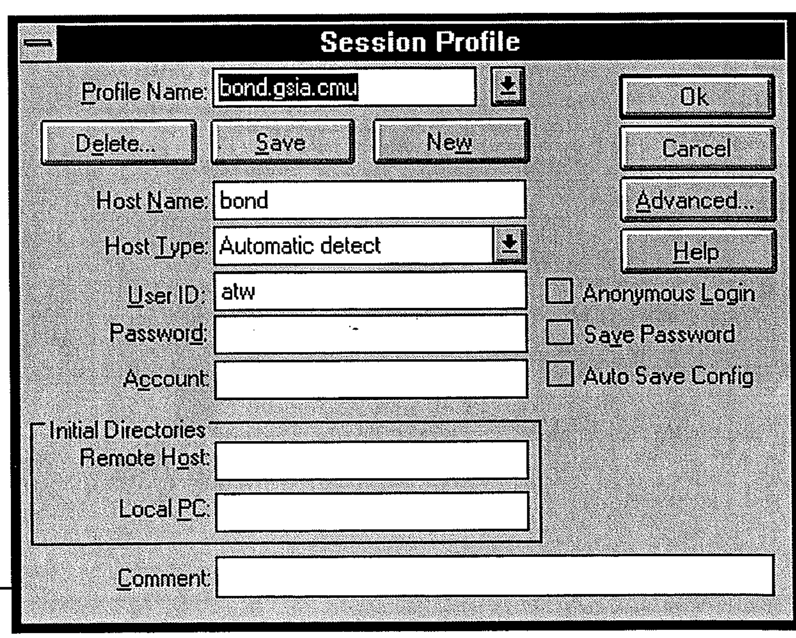
Method 2: using FTP

1. Using the PC, access Program Manager from the Program manager Double-click on the NETWORK icon . A new screen appears

2.Double-click on the FTP icon from the screen



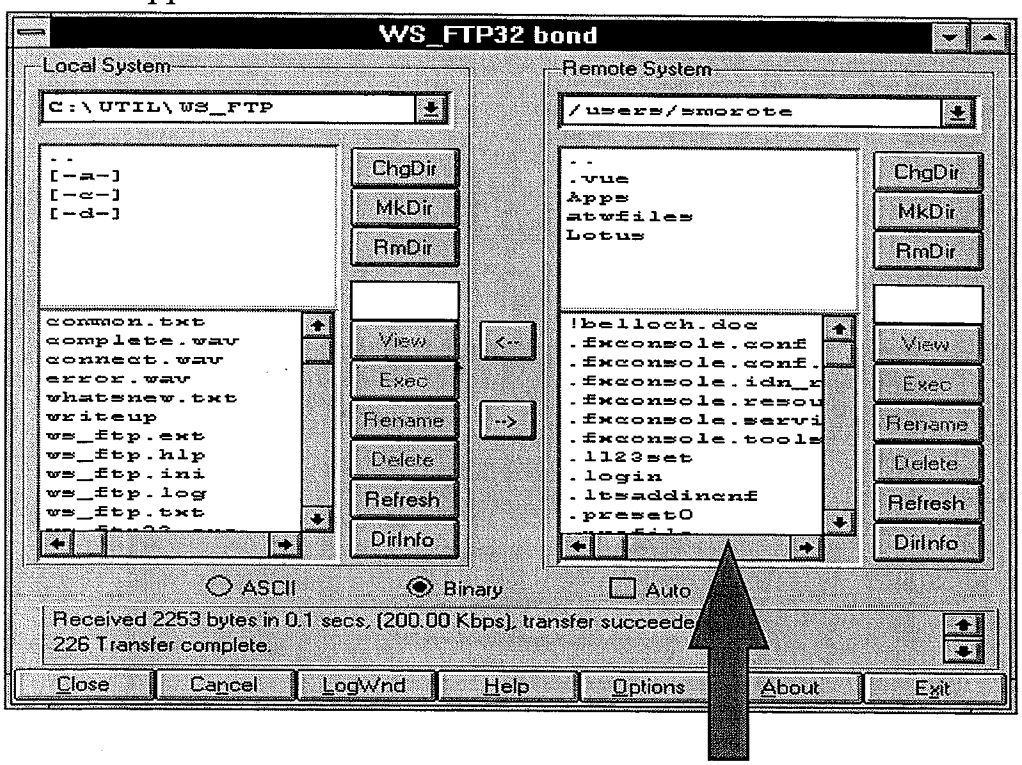
A new window will appear:



Activity 2 (Con't)

In this window, enter the following information
Host name (The name of your HP)
User ID (your atw ID)
Password (your password)
and press "Enter".

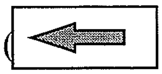
A new screen appears:

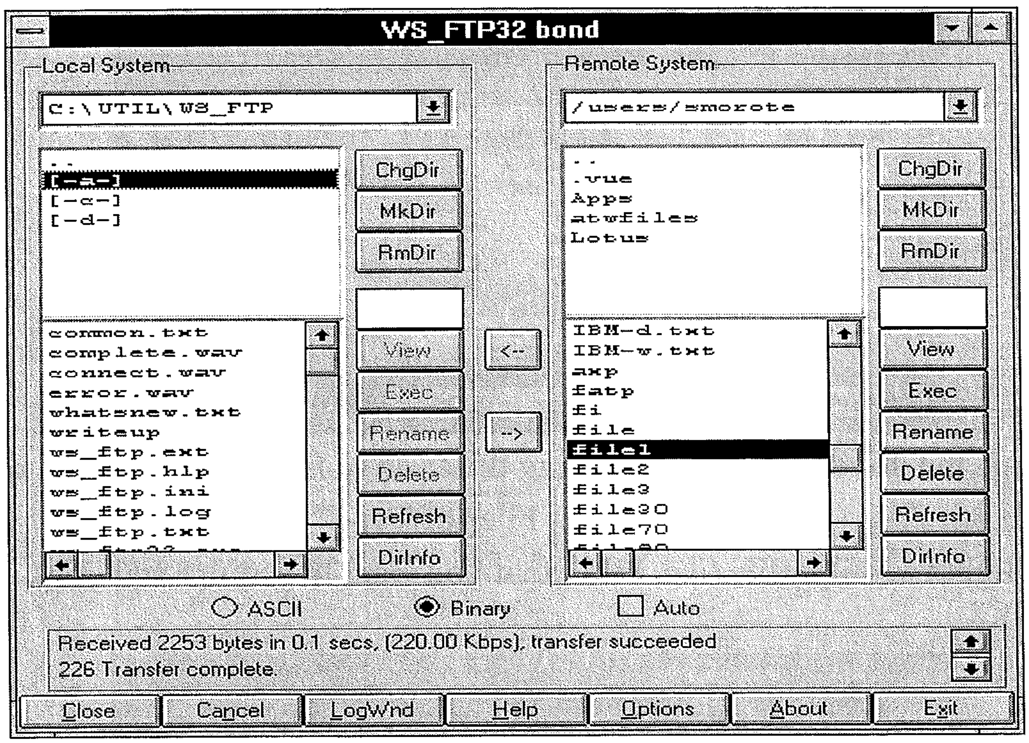


your files

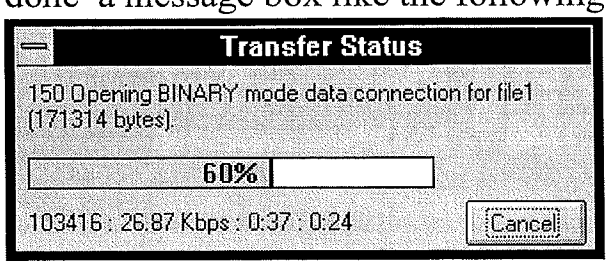
Activity 2 (Con't)

To translate this information change directory: First Click on "a:" and Second click on CHGDIR

3. Select the file you want to translate (e.g., file1) from the remote system list .
4.. Click on the left pointing arrow key () (located in the center of the window) this will begin the translation.



As the translation is done a message box like the following will appear.

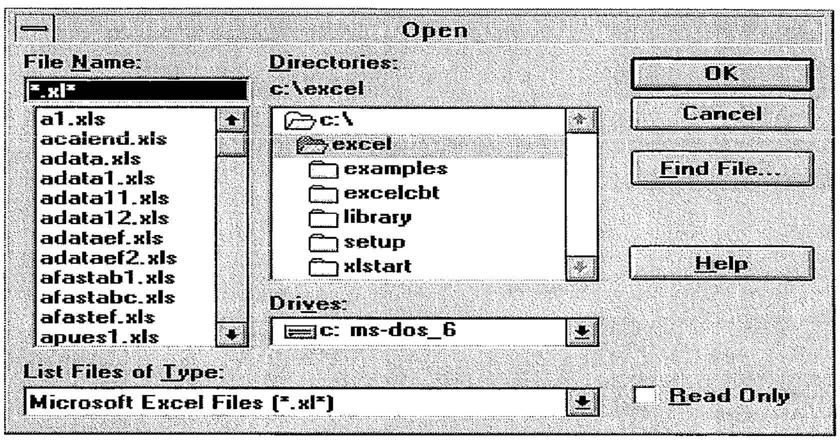


Activity 3: Import the Data into Excel spreadsheet

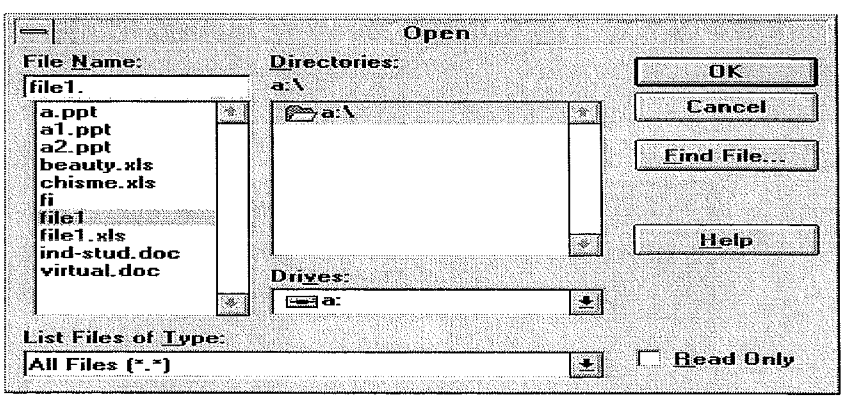
In this simple activity open the outputfile to enter data into Excel.

1. Double click on ICON EXCEL
2. Click on FILE from the menu and Select OPEN from the pull-down menu

A dialogue box will appear:

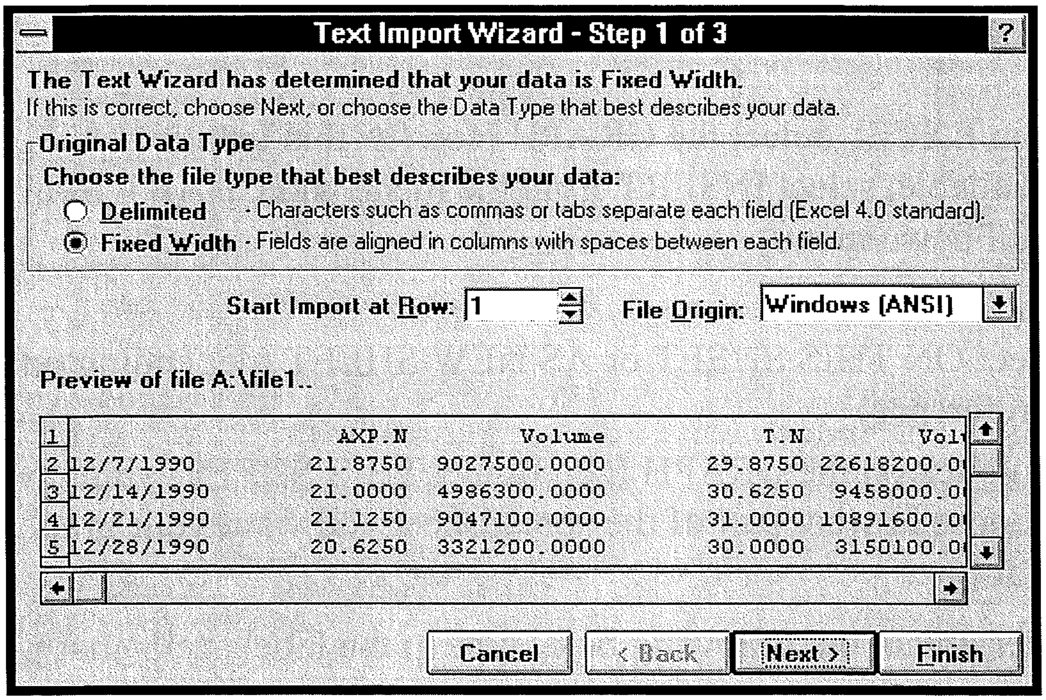


3. Change the drive to "a:" by selecting the a: drive from the " Drives" box
4. Select All Files (*.*) from the "List Files of Type" box
5. Highlight the outputfile (e.g. File1)that you want to acquire and click on "OK"



Activity 3 (con't)

The following "Text Import Wizard" window appears.



6. Click on "Next" in the lower right portion of the window.
7. Click on "Finish" in the lower right portion of the window.
8. The data will now appear in an Excel spread sheet.

Activity 4: Plot the prices as a function of time

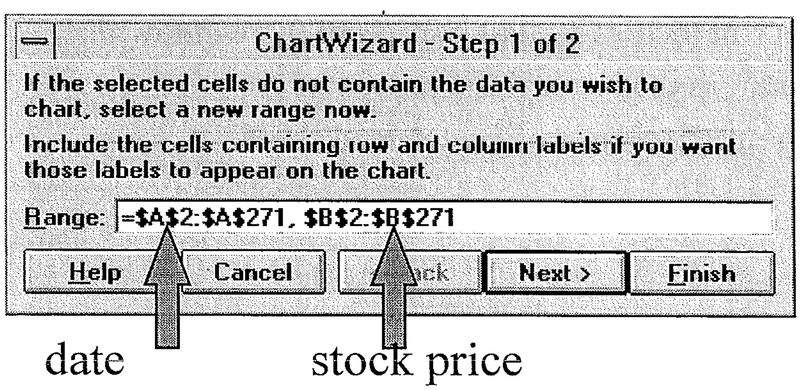
- **Plot the prices on EXCEL** :

1. From EXCEL select the GRAPH Icon (If this Icon is not available, select INSERT from the menu and CHART from the pull down menu)

2. Select ON THIS SHEET or AS NEW SHEET (depending on your preference)

If you click on the GRAPH Icon or ON THIS SHEET, you have to choose in what part of the spreadsheet you want your graph.

3. After obtain the following box, type in the labels cells that you want to plot



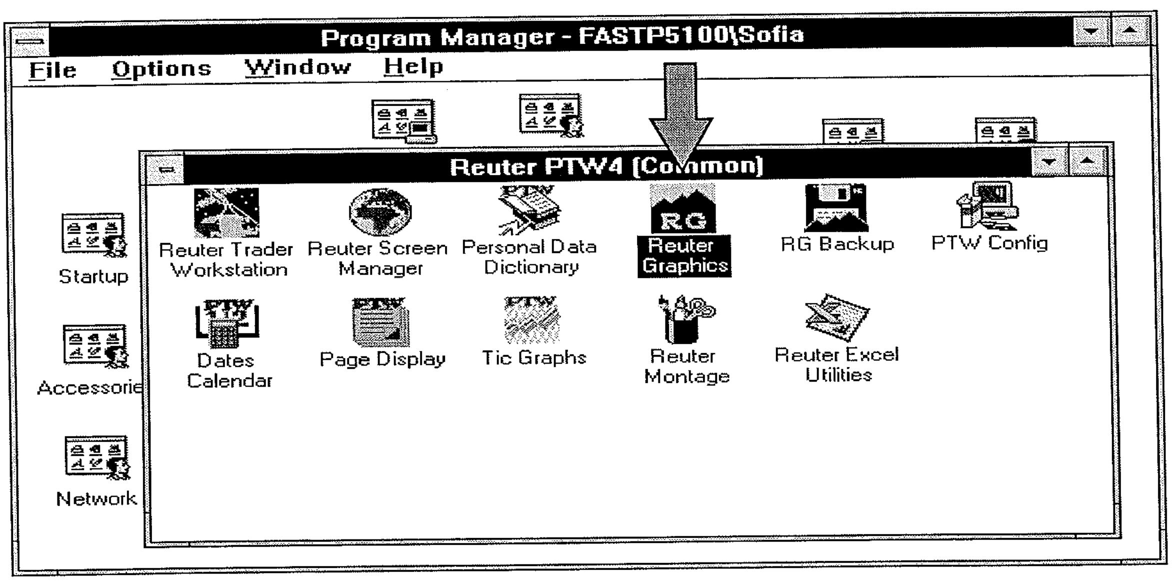
Activity 4 (con't)

Decided what kind of graph you would like using the available options.

Suggestion

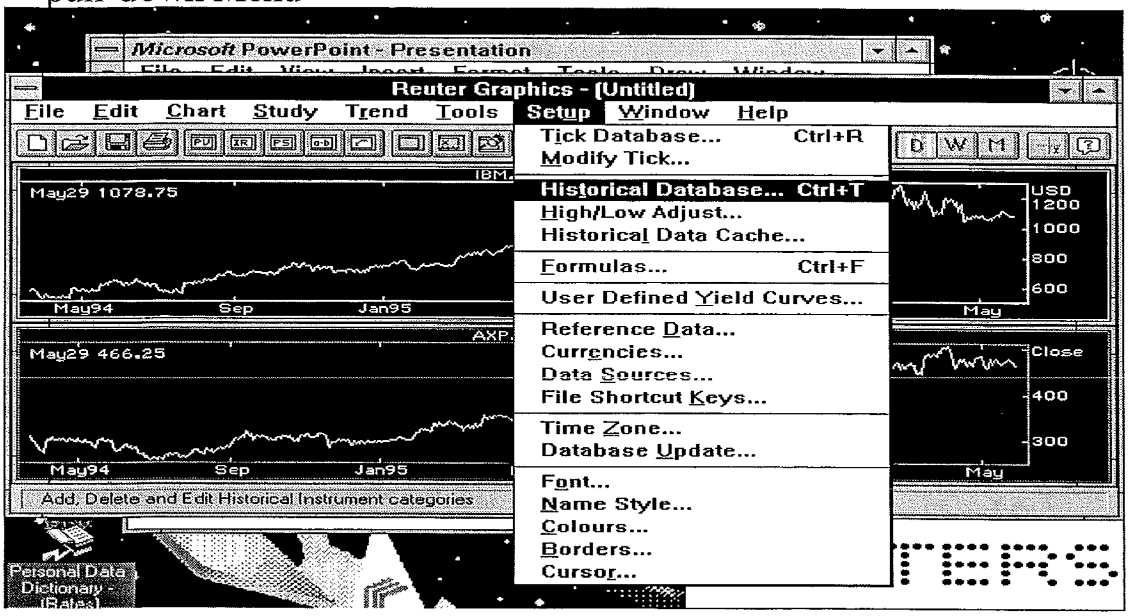
- To quickly plot the price as a function of time, from PTW:

1. Double-Click on PTW REUTER GRAPHICS



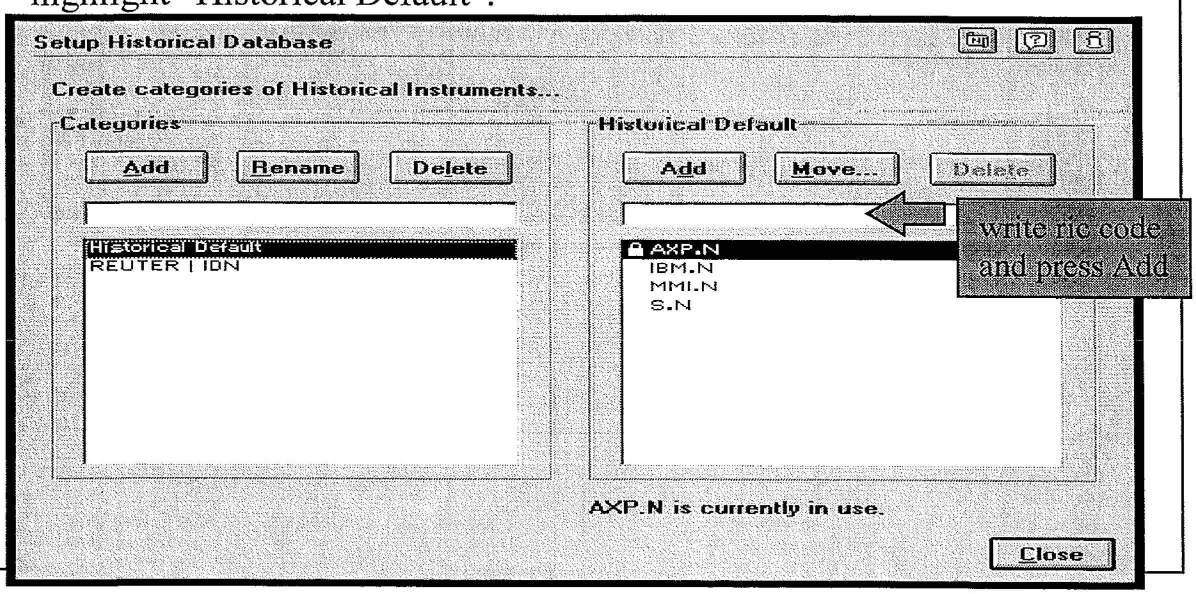
Activity 4 (con't)

2. Select SETUP from the menu bar and HISTORICAL DATABASE from the pull-down Menu



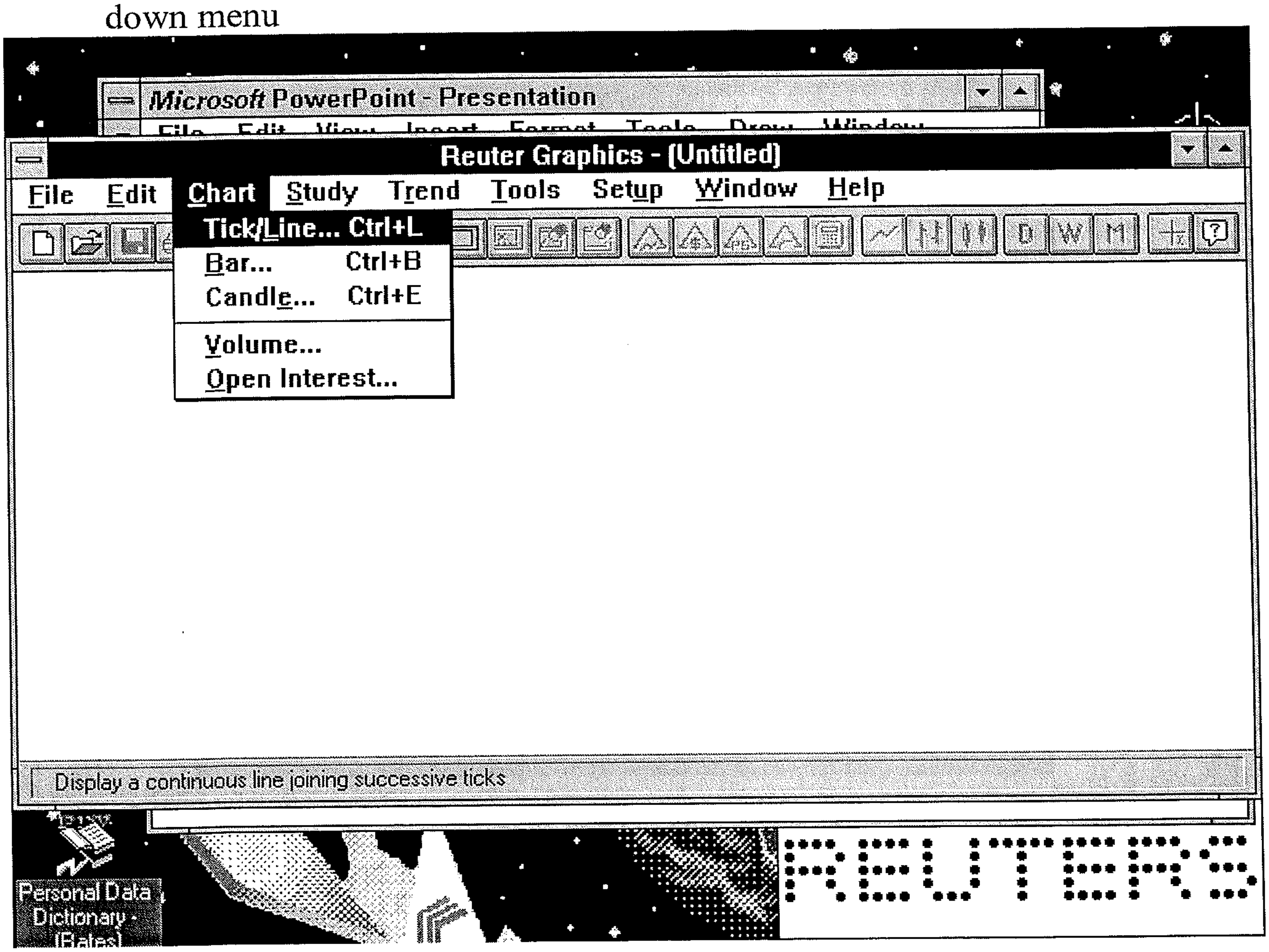
A dialogue box appears

3. Fill in the Ric code of the 20 stocks in the right box. In the left box, highlight "Historical Default".



Activity 4 (con't)

4. Now Select CHART from the menu and Select TICK/LINE from the pull-down menu



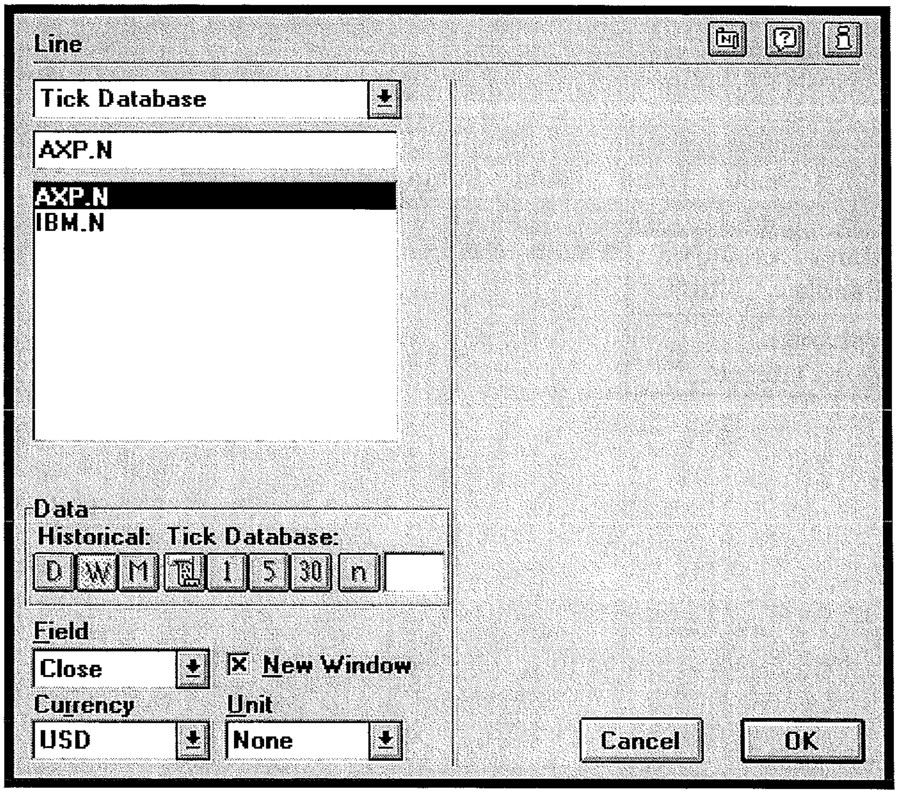
A box will appears with the name of the RICS you just entered.

5. Click on the name of the stocks you want to plot

6. Click on " W" (for weekly)

7. Click the box next to " New Window"

Activity 4 (con't)



8. Press OK .

A graph will appear. Repeat this procedure for each of the 20 stocks.

Activity 5: Calculate the average return on each stock and the standard deviation of returns, and Then compare these to the average return and standard deviation of the S&P500.

- To **calculate** the average return and standard deviation, you can use the following commands:

  =AVERAGE(....)

  =STDEV(.....)

where (....) is the column includes the data for a particular stock.

- **Comments**:

  The Standard & Poor's 500 (S&P 500) Index trades on Chicago Mercantile Exchange (CME) and is based on a portfolio of 500 different Stocks: 400 industrials, 40 utilities, 20 transportation companies and 40 Financial institutions.

Macros Excel

Suggestion

Try to use EXCEL MACROS for "automatic repeated tasks"

Create a Macro

1. From the "Tools" menu , choose "Record macro".
2. Choose "Record new macro"
3. From the Macro box, type in the macro name you want to use.
4. Click "OK"
5. Perform all the activities you want to save with the macro. When you are done, Click on "Stop"

Running a Macro

1. From the "Tools" menu, choose "Macro"
2. From the "Macro Name/reference box", type or select a macro name.
3. Choose the Run button.

Financial Data Lab

Assignment 3

Sofia Morote

1. Find the RIC codes for the following US Treasury Instruments:

i. A Treasury bill maturing in 1 week
ii. A Treasury bill maturing in 1 month
iii. A Treasury bill maturing in 3 months
iv. A Treasury bill maturing in 6 months
v. A Treasury note maturing in 1 year
vi. A Treasury note maturing in 3 years
vii. A Treasury note maturing in 5 years
viii. A Treasury bond maturing in 10 years
ix. A Treasury bond maturing in 15 years
x. A Treasury bond maturing in 20 years
xi. A Treasury bond maturing in 30 years

2. For each instrument, find the exact maturity date, and also yield to maturity shown on the PTW.
3. Link the RIC codes into Excel spreadsheet and plot the yields to a maturity as a function of time
4. Construct the zero-coupon yield curve for the first 6 bonds using a bootstrap procedure(Method 3 in the reading "Constructing Your Own Yield Curve").

Activity 1: Find the RIC codes.

ON LINE DIRECTORY

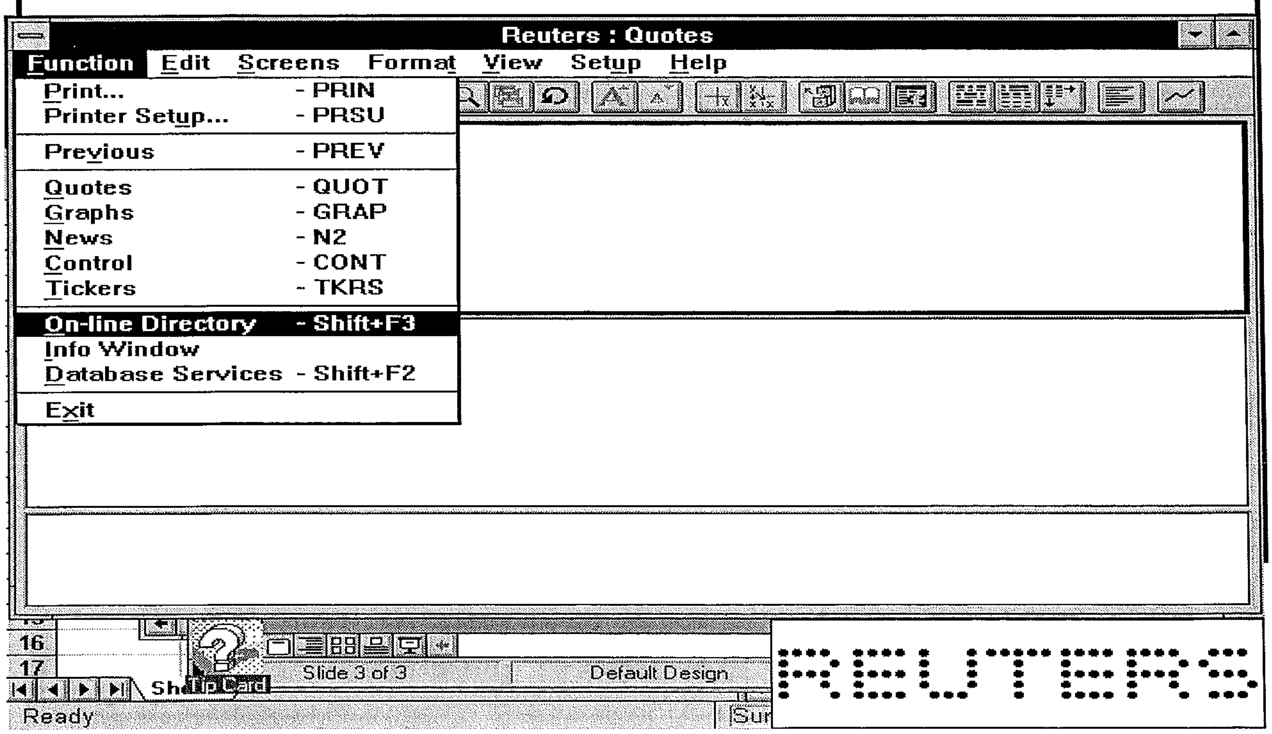
The Reuter Online Directory provides users with a simple method of finding Reuter Instrument Codes (RICs). The directory screen provides information on how to find a RIC. This option is permissionable and may **not** be available to all users in all areas. It may also not be available if your receive the Reuter Terminal (RT) service via broadcast delivery.

How to access instructions for viewing the Reuter Online Directory:

1. Use a mouse, click on Viewing the Reuter Online Directory from PTW

(If using a keyboard, press the tab key to highlight Function).

2. Tab down to “On-line Directory” and press “Enter”.



Activity 1 (con't)

If you don't have data available in the ON LINE DIRECTORY, use the TREASURY 2000 DIRECTORY.

Using the TREASURY 2000 DIRECTORY, you can find the following RIC codes for the assignment:

Treasury Bills	BLBC!
Treasury Notes	NTBC!
Treasury Bonds	BDBC!

USA

US3MT=	3 Month Bill
US6MT=	6 Month Bill
US1YT=	1 Year Bill
US2YT=	2 Year Note
US3YT=	3 Year Note
US5YT=	5 Year Note
US7YT=	7 Year Note
US10YT=	10 Year Bond
US30YT=	30 Year Bond

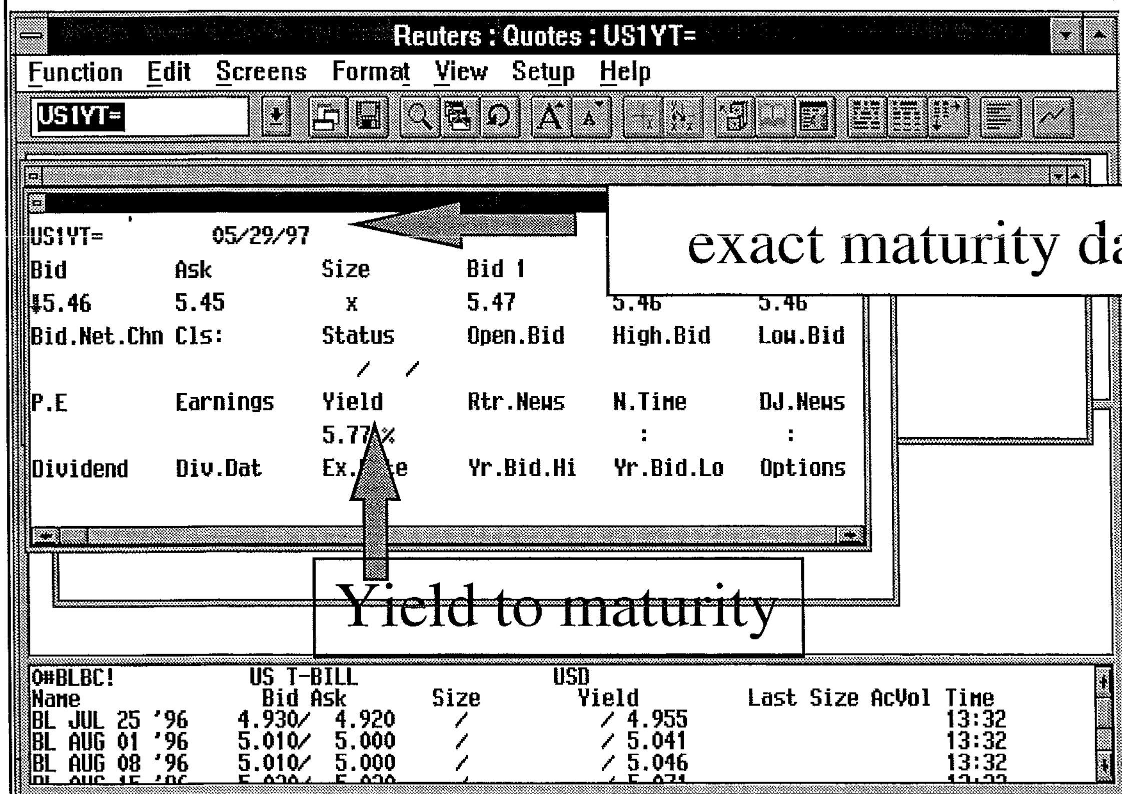
Note: To Find other RIC codes, see example 2 of activity 2.

Activity 2

Example:

Treasury Bill maturing 1 year:

Using PTW, Fill the RIC Code and press enter

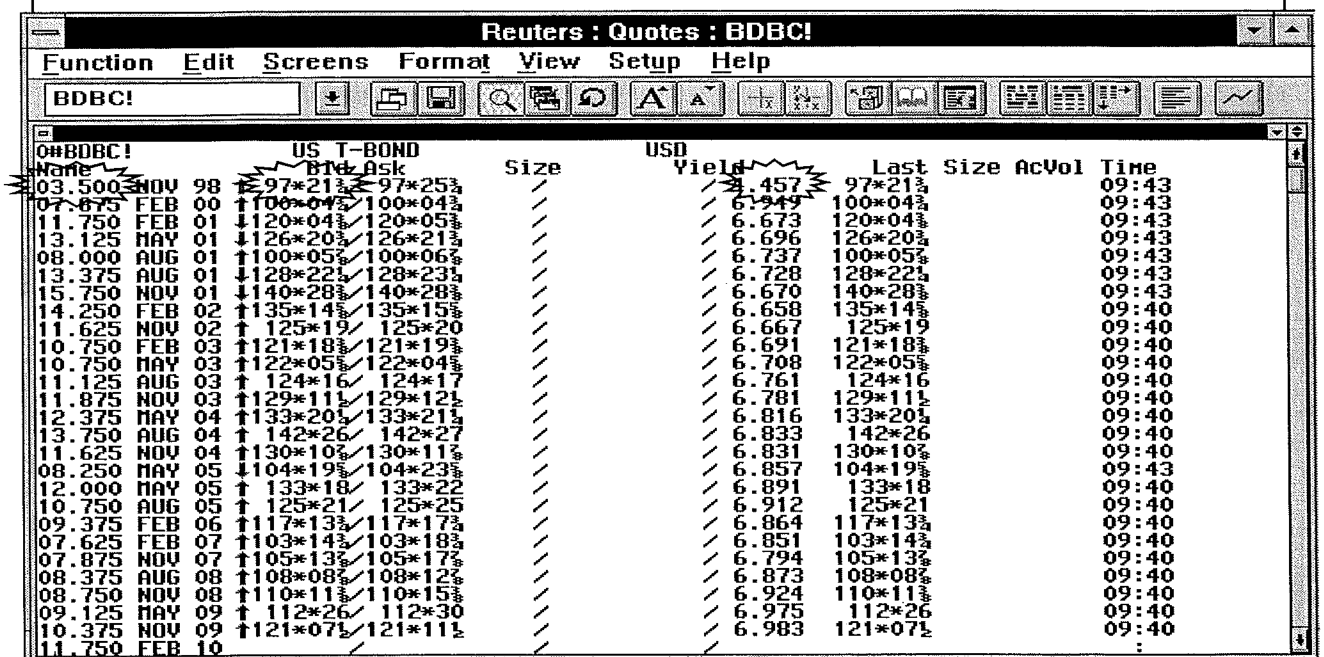


Activity 2 (con't)

Another example

To find information about a Treasury Note 38 Year Note matures in 2.5 year, do the following:

1. Using PTW type "BDBC!" in the box (directly beneath the world "Function") and press F3.



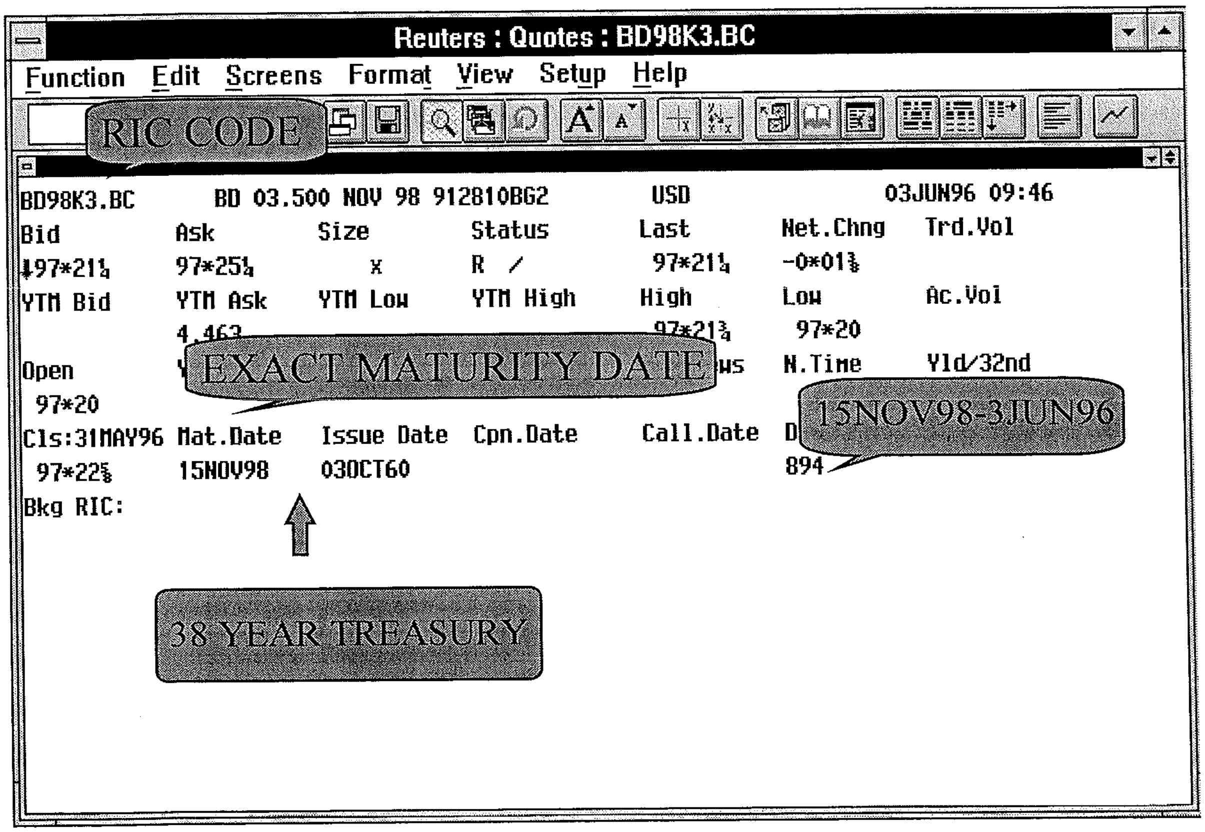
0#BDBC!		US T-BOND		USD				
Name		Bid	Ask	Size	Yield	Last	Size	AcVol	Time
03.500 NOV 98	↑	97*21¾	97*25¾	/	4.457	97*21¾			09:43
07.875 FEB 00	↑	100*04¾	100*04¾	/	6.949	100*04¾			09:43
11.750 FEB 01	↓	120*04⅜	120*05⅜	/	6.673	120*04⅜			09:43
13.125 MAY 01	↓	126*20¾	126*21¾	/	6.696	126*20¾			09:43
08.000 AUG 01	↑	100*05⅞	100*06⅞	/	6.737	100*05⅞			09:43
13.375 AUG 01	↓	128*22¼	128*23¼	/	6.728	128*22¼			09:43
15.750 NOV 01	↓	140*28⅜	140*28⅜	/	6.670	140*28⅜			09:43
14.250 FEB 02	↑	135*14⅝	135*15⅝	/	6.658	135*14⅝			09:40
11.625 NOV 02	↑	125*19	125*20	/	6.667	125*19			09:40
10.750 FEB 03	↑	121*18⅜	121*19⅜	/	6.691	121*18⅜			09:40
10.750 MAY 03	↑	122*05⅝	122*04⅝	/	6.708	122*05⅝			09:40
11.125 AUG 03	↑	124*16	124*17	/	6.761	124*16			09:40
11.875 NOV 03	↑	129*11½	129*12½	/	6.781	129*11½			09:40
12.375 MAY 04	↑	133*20¼	133*21¼	/	6.816	133*20¼			09:40
13.750 AUG 04	↑	142*26	142*27	/	6.833	142*26			09:40
11.625 NOV 04	↑	130*10⅞	130*11⅞	/	6.831	130*10⅞			09:40
08.250 MAY 05	↓	104*19⅝	104*23⅝	/	6.857	104*19⅝			09:43
12.000 MAY 05	↑	133*18	133*22	/	6.891	133*18			09:40
10.750 AUG 05	↑	125*21	125*25	/	6.912	125*21			09:40
09.375 FEB 06	↑	117*13¾	117*17¾	/	6.864	117*13¾			09:40
07.625 FEB 07	↑	103*14¾	103*18¾	/	6.851	103*14¾			09:40
07.875 NOV 07	↑	105*13⅞	105*17⅞	/	6.794	105*13⅞			09:40
08.375 AUG 08	↑	108*08⅞	108*12⅞	/	6.873	108*08⅞			09:40
08.750 NOV 08	↑	110*11⅜	110*15⅜	/	6.924	110*11⅜			09:40
09.125 MAY 09	↑	112*26	112*30	/	6.975	112*26			09:40
10.375 NOV 09	↑	121*07½	121*11½	/	6.983	121*07½			09:40
11.750 FEB 10			/	/					:

COUPON RATES

For e.g. Nov 98, the data shows a coupon rate of 3.5 % YTM 4.457% and bid 97+(21.75)/32 = 97.68

Activity 2 (con't)

- By double-clicking on Nov 98 you will get specific information is available:



Activities 3 and 4

Activity 3: Link the RIC codes into Excel spreadsheet and plot the yields to a maturity as a function of time

Complete Activity 3 by following the instructions (about how to link) given in Assignment 2.

Activity 4: Construct the zero-coupon yield curve for the first 6 bonds using a bootstrap procedure(Method 3 in the reading "Constructing Your Own Yield Curve").

The Zero-coupon yield curve is a curve showing the relationship between spot rates (i.e., zero-coupon yields) and maturity.

DETERMINATION OF ZERO-COUPON YIELD CURVE

In practice, spot rates (or zero-coupon yields) cannot always be observed directly. What can be observed are the prices of coupon-bearing bonds. An important issue, therefore, is how the zero-coupon yield curve can be extracted from the price of the coupon

One commonly used approach is known as the bootstrap method

Activity 4 (con't)

In this activity, use Method 3 from the reading entitled "constructing your own yield curve" to construct a yield curve

Example in "constructing your own yield curve"
Suppose we have:

Maturity	1+spot rate
1 year	1.0400
2 year	1.0696

The value of a 2 year coupon bond (coupon rate of 5 % compounded annually with face value equal to $100):

$$\frac{5}{1.04} + \frac{105}{1.0696^2} = \$96.587$$

Now the price of a 2 year coupon bond equals 96.587 and the cash flow is $5 at the end of year 1 and $105 at the end of year 2.

(Note that the correct numbers have been included in this example in the calculation)

Activity 4 (con't)

Example of bootstrap method:

Suppose you have:

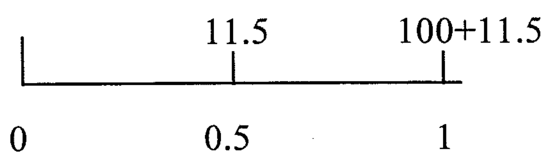
6 months T-Bill 5.382 yield

and suppose a Bond where:

Ask Price = 101.90 (Ask price is usually used)

Quoted Rate: 11.50%

Also suppose you want to construct a zero coupon yield curve for the one-year bond using the bootstrap procedure:



We can obtain the value of the yield to maturity by using the following relation:

$$101.90 = \frac{11.5}{(1+5.382/100)^{0.5}} + \frac{111.5}{(1+YM/100)^{1}}$$

Hull suggests the following equation (John Hull: Options, futures and other Derivative Securities: pages 84-87)

$$101.90 = 11.5 * e^{(-5.382/100)*0.5} + 111.5 * e^{(-YM/100)*1}$$

Notes:

NORMAL SHAPE

- The yield curve is normally positively sloped (long bonds yield more than short bonds, because long bonds are more risky). Occasionally, when monetary policy is very tight, the yield curve is inverted and short bonds yield more than long ones.

YIELD CURVE JARGON

- Positively sloped (normal): long rates higher than short.
- Inverted: short rates higher than long.
- Humped: medium rates higher than both short and long.
- Yield curve steeping: increasing spread between short rates and long rates, e.g. long rates rising faster than short.
- Yield curve flattening: narrowing spread between short rates and long rates, e.g. long rates falling faster than short.

Notes

Normal Yield curve

Inverted Yield curve

Humped yield curve

Yield Curve Steeping

Yield Curve Flattening

Financial Data Lab
Assignment 4
Sofia Morote

The Black-Scholes Option Pricing formula is

$$C = SN(d1)-X\exp\{-rt\}N(d2)$$

Where S = stock price, etc.

Often, we use the formula to calculate the implied volatility, i.e. the volatility with which the Black-Scholes formula holds exactly.

Find the quotes on option prices and build a real-time calculator which takes as an input the stock price on AT&T (ticker: T.N). For each strike price and maturity calculate the implied volatility calculate the implied volatility by writing a visual basic macro which implements the bisection method.

Calculate the implied volatility for 5 AT&T strike prices by writing a visual basic macro which implements the bisection method

INDEX

ACTIVITIES

Enter to the PTW

1. Type the Equity Option Chain Code.

T*.W

2. Press the CHAIN function key - F3.

Using PTW and Excel, link the call(Put), Stock value, Strike price and date into spreadsheet so they update automatically.

EXCEL

1.Calculate the implied volatility by writing a visual basic macro which implements the bisection method.

2.Check your results:
Input-values:
S = 62
X = 65
r = 0.1
T-t = 0.5
c = 2.75

Left sigma = 0.001
Right sigma = 2
Accuracy = 0.0001

you will obtain:

0.152597 as implied volatility

THE BLACK-SCHOLES OPTION FORMULA

The key element of any option value model is the probability assumptions about changes in underlying asset price. If the probability distributions of asset price changes are known or can be successfully estimated, then these may be used to derive the probability densities of the expected payoff schedule of options at expiration, from which fair value may be derived.

The probability of asset price change is usually referred to as volatility by option traders, and it is usually unknown. As a consequence, **volatility** must be estimated. Doing so gives rise to a number of different option pricing models.

Two other statistical assumptions or unknowns must also be incorporated in order to derive fair value for any model.

(1) The risk-free interest rate must be known or estimated over the life of the option; and,

(2) if the option is on a yield-bearing asset, the dividend or yield must be known or estimated over the life of the option. For bond options in particular, additional estimates of the term structure of interest rates many also be necessary.

The work of Black and Scholes, proposed to link the probability of stock price changes to stock options using a log-normal distribution as the probability estimator. They obtained the exact formulas for the prices of European call and put options.

$$C = S\,N(d1) - X\,e^{-r(T-t)}\,N(d2)$$
$$P = X\,e^{-r(T-t)}\,N(-d2) - S\,N(-d1)$$

where

$$d1 = \frac{\ln(S/X) + \left(r + \frac{sigma^2}{2}\right)(T-t)}{(sigma)\,(T-t)^{0.5}}$$

$$d2 = d1 - sigma\,(T-t)^{0.5}$$

N(d) = cumulative normal integral
r = risk-free interest rate
sigma = standard deviation of log percentaje change in annualized prices
X = strike price
S = Stock price
T-t = time to expiration
C = call premium
P = Put premium

BISECTION METHOD

In many applications, it is necessary to solve an equation of the form

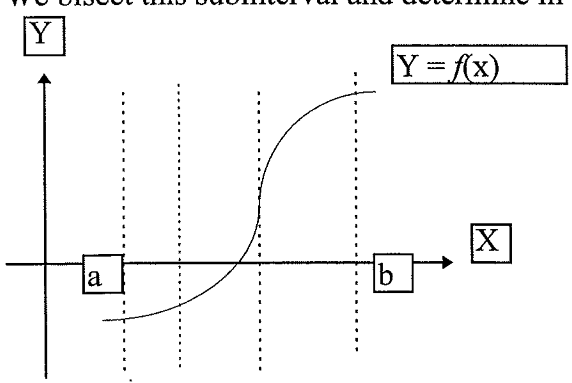
$$f(x) = 0$$

For some functions f, it may be very difficult or even impossible to find an exact solution. Examples include the equation

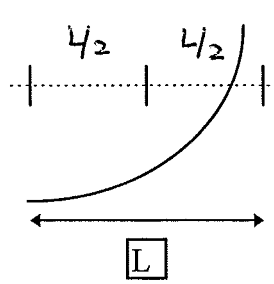
$$A - P\,\frac{(1+r)^n - 1}{r(1+r)^n} = 0$$

which can be solved to find the monthly interest rate r for a loan amount A, where P is the payment to be made for a period of n months.

For such equations, a repetitive numerical method may be used to find an approximate solution. One such method is the **bisection method.** In this method, we begin with two numbers a and b, where the function values $f(a)$ and $f(b)$ have opposite signs. If f is continuous between $x = a$ and $x = b$ then the graph of f must cross the x-axis at least once between $x = a$ and $x = b$; thus, there must be at least one solution of the equation $f(x) = 0$ between a and b. To locate one of these solutions, we first bisect the interval $[a, b]$ and determine in which half f changes sign, thereby locating a smaller subinterval containing a solution of the equation. We bisect this subinterval and determine in which half of it f changes sign;



Repeating this process gives a sequence of subintervals, each of which contains a solution of the equation and has a length once-half that of the preceding interval. Note that at each step, the midpoint of a subinterval of length L is within $L/2$ of the exact solution.



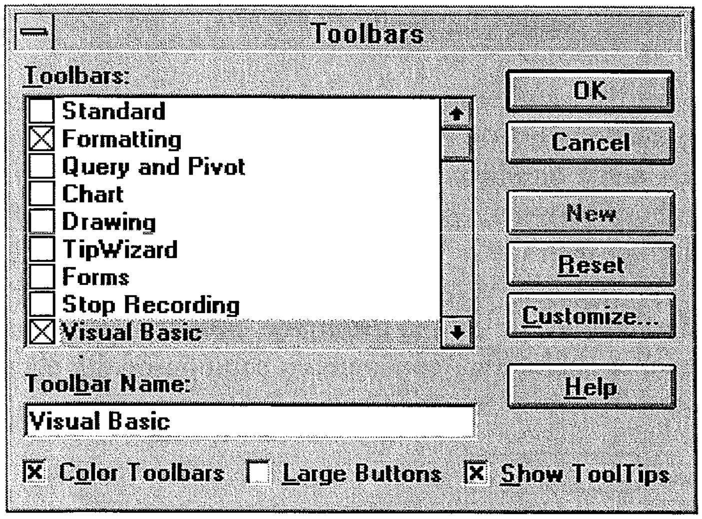
EXCEL PROGRAMMING WITH VISUAL BASIC FOR APPLICATIONS (VBA)

Introducing the Visual Basic Toolbar

If Excel doesn't display the Visual Basic toolbar when you display a module sheet, use the View | Toolbars command to display it. To display the Visual Basic toolbar, follow these steps:

1. Choose the View | Toolbars command. Excel displays the Toolbars dialog box.

2. Select the check box to the left of the Visual Basic toolbar name in the Toolbars list.



3. Choose OK. Excel closes the Toolbars dialog box and displays the Visual Basic toolbar in its last visible position.

Tip: You can also display the Toolbars dialog box by right-clicking (clicking the right mouse button) on any visible toolbar and then choosing Toolbars from the resulting pop-up menu.

Writing the Macro Text.

When you write a macro, you must specify the **macro's name** and include the `sub` keyword at the beginning of the macro and the `End Sub` keywords at the end of the macro. If you omit any of these three elements, the syntax of your macro will not be correct.

The classic first program in any programming language is a program that displays the message *Hello, World!* On the screen. Listing 1 shows just such a VBA macro program.
To enter this macro program yourself, follow these steps:

1. Open any Excel workbook, o create a new workbook.

2. Choose the **I**nsert | **M**acro | **M**odule command to insert a new module sheet in the workbook. Excel inserts the new module and makes it the active sheet.

3. Choose the **E**dit | **S**heet | **R**ename command to display the Rename Sheet dialog box.

4. Enter the name **FirstProgram** in the **N**ame text box.

5. Choose OK. Excel renames the new module sheet.

6. Make sure the insertion point is at the beginning of a blank line, and type the text shown in Listing 1, pressing Enter at the end of each line to start a new line.

Type the source code from Listing 1 into the module sheet exactly as it appears in the listing, but without the line numbers.

Listing 1. The `HelloMacro procedure.`

```
Sub HelloMacro()
    Dim HelloMsg As String
    MsgBox "Hello World", , "Greeting Box"
End Sub
```

Run a Macro

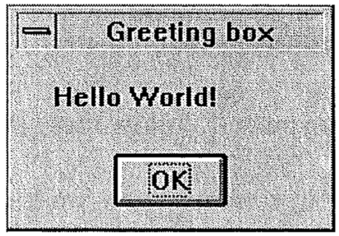
After you enter the source code for the **`HelloMacro`** sub procedure, you can run the macro

1. Choose the Tools | Macro command to display the Macro dialog box.

2. Select the **`HelloMacro`** procedure in the Macro Name/ Reference list.

3. Choose the Run command button.

When VBA executes the **`HelloMacro`** sub procedure from Listing 1, it displays the dialog box shown in next Figure Choose the OK button to clear the dialog box and end the macro.



Using Dim to Declare Typed Variables

To declare a variable and its type with the **Dim** statement, add the keyword As after the variable name, and then type the name of the data type that you want the variable to have. The general syntax to use the Dim statement to declare a typed variable is:

Dim varname As type

varname represents any valid VBA variable name, and *type* represents any one of VBA's data type names. (Table 1 List the names of useful of VBA's data types.)

The following lines show examples of the correct syntax for typed variable declarations:

```
Dim PcntProfit As Single
Dim Gross_Sales As Currency
Dim PayDay as Date
Dim Message As String
Dim Counter As Integer.
```

Using Msgbox

When you use named arguments, you don't have to include place-holding commas for optional arguments. Notice, in the second statement on the preceding page, that there is no place-holder comma between the arguments for the prompt and title, as there is in the first statement. In fact, named arguments do not have to appear in any particular order. In the second statement, you could list the Title argument before thc prompt argument, as shown bclow:

```
MsgBox Title:=AnyTitle, prompt:=AnyMsg
```

MsgBox still uses the value assigned to the Title argument as the dialog box title. When you use named arguments, VBA uses the name of the argument to determine what value that argument represents.

Using Excel's Functions

In addition to the functions built into Visual Basic for Applications, Excel also makes some of its functions available to VBA (see table 2 in the appendix). Excel has a wide variety of functions that perform mathematical, logical, financial, and statistical operations on data in worksheets. Excel makes many, but not all, of these functions available to VBA.

The functions that Excel makes available to VBA are not a part of VBA. They are part of Excel.

To use a function that belongs to Excel (or any host application), you access the function in VBA through the `application` program object. The application Object in VBA represents the host `application` and all its resources.

The following statement use the Excel Max function, which returns the largest number in its argument list:

```
MsgBox Application.Max(4, 1, 3, 2)          'Displays 4
```

In this statement, notice that the keyword application is followed by a period(.) and then the name of the function, Max without any spaces. The period–called a *dot separator*–indicates that the statement refers to the Max function, which is part of the application object. When you use Excel's functions in your VBA macro programs, you must include the Application object. When you use Excel's functions in your VBA macro programs, your must include the application keyword and the dot separator (.) in front of every Excel function name.

Using InputBox

The InputBox function also has named arguments, as do all of the VBA functions. The following statement shows as InputBox statement that uses named arguments:

```
Name =Application.InputBox(prompt:=AnyTEx,Title:=AnyTitle)
```

Notice that this statement includes parentheses around the argument list. Your must always include parentheses around the argument list when you use a function's result, whether or not you use named arguments when you call the function.

Using the Range Method

You can use the Range method to return a rectangular range of cells. The following example fills the range A1:H8 with the string "Test."

```
Sub FillTheRange()
```

```
    Range(Cells(1, 1) Cells (8, 8)). Value = "Test"
End Sub
```

The Range method provides additional shortcuts for referring to ranges of cells in very simple procedures. For example, **`Range("A1") = 24`** sets the value of cell a1 to 24. For more information about the **Range** method, search the on-line Visual Basic Reference for *Range method.*

Writing a Function Procedure.

Function procedures are very similar to the VBA procedures you already know how to write. The main difference between a function procedure and other procedures (apart from the fact that functions return a value and procedures do not) is that you enclose a function procedure with the keywords `Function` and `End Function` instead of the `Sub` and `End Sub` keywords you are already familiar with.

The general syntax for a function procedure is:

```
Function name([arglist])
      'VBA Statements
      [name =  expression]
End Function
```

Every function procedure begin with the restricted keyword function, followed by the name of the function procedure. `Name` represents the name you choose for the function procedure. Function names must follow the same rules as any other identifier name in VBA: they must begin with a letter, may not contain spaces or any of the arithmetic, logical, or relational operator symbols, and may not duplicate any of VBA's restricted keywords.

Example

The **If...Then...Else** statement you can see how it works in the Power function. Listing 2 shows a function, named Power, that returns the power of a number, given the number and the power to raise it to.

'Listing 2 function procedure

```
Function Power(num As Double, pwr As Intenger)
   If pwr = 0 Then
       Power = 1
    Else
        Power = num * Power(num, pwr - 1)
    End If
End Function
```

Line 1 contains the Power function declaration. Power has two required arguments. Incidentally, you don't really need a write a Power function, the VBA exponentiation operator (^) has the same effect.

You **can use now**, you power function, writing in any cell of your spreadsheet:
= power(value, value)
and press enter
e.g.
= power(2,3)
returns value = 8.

Building Loops with *Do While*

The first loop construction that test its determinant condition before executing the loop is the `Do While`.

The general syntax of the `Do While` statement is:

```
Do While condition
      statements
Loop
```

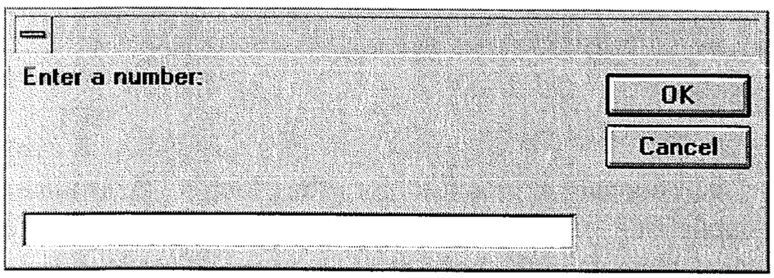
`Condition` represents the logical expression for the loop's determinant. `Statements` represents none, one, or several statements that make up the body of the loop. VBA executes all statements in the body of the loop each time it executes the loop. The `Loop` keyword after statements indicates the end of the loop's body and also indicates the point at which VBA returns to the top of the loop to check the determinant condition.

Example
`Count_OddNums` simply demonstrates how you go about constructing a `Do While` loop. The loop in this procedure executes while the user has entered less than 10 odd numbers. At first, it might seem that you could accomplish this task with a `For...Next loop`, but that is not really possible. Although the loop in this procedure is a count-controlled loop, the counter that controls the loop's execution is not necessarily incremented every time the loop executes. If the user enters an even number, the loop does not increment the count of odd numbers. The only way to create a count-controlled loop that increments the c counter variable at irregular intervals is with some variety of the `Do` statement.

```
' Listing 3 Count_OddNums()
Sub Count_OddNums()

Dim OddCount As Integer
Dim OddStr As String
```

```
Dim Num
OddStr = " "
OddCount = 0
Do While OddCount < 10
   Num = InputBox("Enter a number:")
   If (Num Mod 2) <> 0 Then
      OddCount = OddCount + 1
      OddStr = OddStr & Num & " "
   End If
Loop

MsgBox prompt:="You entered the following odd " & _
               "numbers: " & Chr(13) & OddStr, _
            Title:=ocTitle
End Sub
```

When you run this program a inputbox appear:



Using the For...Next Loop

The first of VBA's For Loops is the For...Next loop. Use the For...Next loop when you want to repeat an action or series of actions a set number of times.

The `For...Next` loop has the following general syntax:

```
For counter = start To end [Step StepSize]
      statements
Next [counter]
```

`counter` represents any VBA numeric variable, usually an Integer or Long type variable, `start` represents any numeric expression, and specifies the starting value for the counter variable. `end` is also a numeric expression and specifies the ending value for the counter variable.

What is a Visual Basic Procedure?

Now that you're comfortable with macros and user-defined function, you're ready to use the standard Visual Basic term for these words-procedure.

Procedure

macro → Sub Procedure

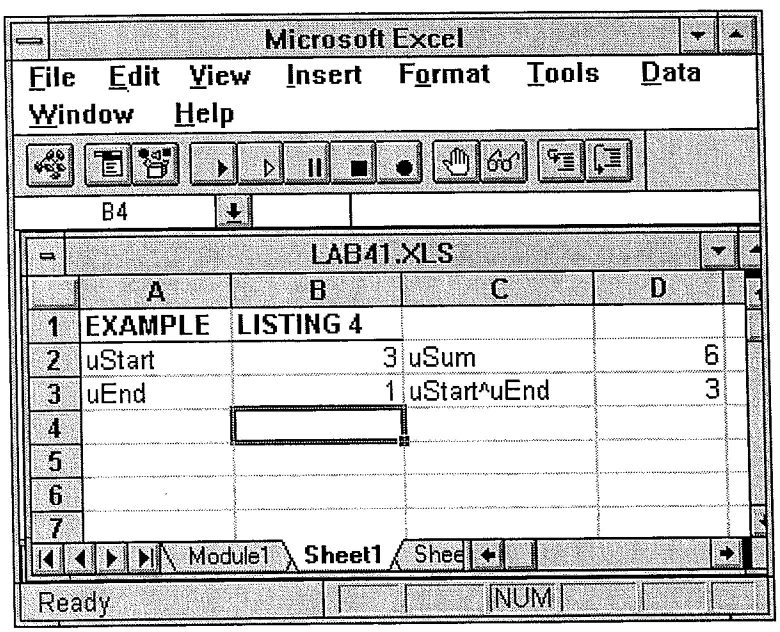
User-Defined function → Function Procedure

```
'Listing 4
'Example use of function and Sub, For next and Range method

Sub Demo_ForNextDown()
   Dim k As Integer
   Dim uStart As Double
   Dim uEnd As Double
   Dim uSum As Long

   uStart = Range("b2") 'using range method
   uEnd = Range("b3")
   uSum = 0

   For k = uStart To uEnd Step -1   'loop counts down
      uSum = uSum + k
   Next k

 MsgBox "The sum of the numbers from " & uStart & _
        " to " & uEnd & " is: " & uSum

Value = Power(uStart, uEnd)  ' Using Function Power, previously defined

        Range("d2") = uSum  'put uSum in cell d2
        Range("d3") = Value
   End Sub
```

When VBA executes statement uEnd = Range("b3"), it first call the cell "B3" of the active spreadsheet to get a number from the user. VBA stores that value in uEnd variable.

Similarly, in the lasts lines also uses Range method to get the value from VBA and return this value in the cells d2 and d3 to the user.

We fill our values uStart and uEnd before run the program. And VBA returns the answers in the cells d2 and d3



Appendix

1. Where to Find Information About Visual Basic

Your Microsoft Excel package includes the following four sources of information about Visual Basic:

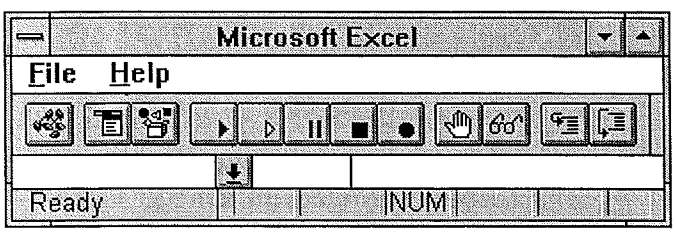
- The *Microsoft Excel Visual Basic User's Guide* (this manual) teaches you how to record, run, and edit macros; how to create and enter user-defined function; and how to program in Visual Basic so that you can write your own macros or even create complete applications.

- The main Microsoft Excel Help file provides general on-line Help about macros (what they are and how to record and run them)

- Examples and Demos are lesson within on-line Microsoft Excel Help that show you how to perform common macros tasks.

The Visual Basic Reference is a detailed on-line reference for Visual Basic function, statements, methods, properties and objects.

2. The Visual Basic toolbar has 12 buttons. Each button's action is summarized below:



- *Insert Module*. This button inserts a module sheet in front of the currently active sheet. Choosing this button is the same as using the **I**nsert | **M**acro | **M**odule command to insert a module sheet.

- *Menu Editor*. This button stars the Menu Editor. Choosing this button is the same as using the **T**ools | Menu E**d**itor command. You use the menu editor to customize your application's menus.

- *Object Browser*. This button stars the Object Browser. Choosing this button is the same as using the **V**iew | **O**bject Browser command. You use the Object Browser to see a list of the macros currently available, among other tasks.

- *Run Macro*. Use this button to run a macro. If the active window displays a module, choosing this button is the same as using the **R**un | **S**tart command. If the active window is not a module, choosing this button has the same effect as using the **T**ools | **M**acro command to display the Macro dialog box.

- *Step Macro*. Use this button when you debug your macros. Choosing this button stars VBA's break mode and displays the Debug window.

- *Resume Macro*. This button resumes the execution of a paused macro; you usually use this button only while debugging macros.

- *Stop Macro*. Use this button to stop a running macro; if the macro recorder is running, this button stops the macro recorder.

- *Record Macro*. This button to start the macro recorder. Choosing this button is the same as using the **T**ools | **M**acro | **R**ecord Macro command to start the macro recorder.

- *Toggle Breakpoint, Instant Watch, Step Into, Step over*. You use these last few buttons on the Visual Basic toolbar only when you are testing and debugging your macros. Usually, you use these buttons in conjunction with the Debug window. The Toggle Breaking button corresponds to the **R**un | Toggle **B**reakpoint command; Instant Watch corresponds to the **T**ools | Instant **W**atch command' and the Step Into and Step Over buttons correspond to the commands of the same name on the **R**un menu.

Table 1. Visual Basic data types.

Type Name	Size in By	Description and Value Range
Array	As required by the type and number of array elements	Each array element's range is the same as the base type. The number of elements in an array has no fixed limit
Boolean	1(16 bits)	Stores logical values; may contain only the values `True or False`.
Long	4(32 bits)	Whole numbers from -2,147,483,648 to 2,147,483,647.
Object	4(32 bits)	Used to access any object recognized by VBA. Stores the memory address of the object.

Single	4(32 bits)	Negative number: from -3.402823×10^{38} to $-1.401298 \times 10^{-45}$. Positive numbers: from 1.401298×10^{-45} to 3.402823×10^{38}.

Table 2 VBA's mathematical functions.

Function(Arguments)	**Return/Action**
`Abs(N)`	Returns the absolute value of `N`.
`Exp(N)`	Returns the constant *e* raised to the power `N`. (*e* is the base of natural logarithms, and is approximately equal to 2.718282.)
`Log(N)`	Returns natural logarithm of `N`.
`Rnd(N)`	Returns a random number; argument is optional. `Rnd` is used to provide a random factor in programs that simulate some real-word event, such as stock market simulations. Use the `Rnd` function only after initializing VBA's random number generator with the `Randomineze` statement.
`Sqr(N)`	Return the square root of `N`. VBA displays a runtime error if `N` is negative. (By mathematical definition, negative numbers cannot have a square root.)

3. Understanding Error Messages While Writing, Editing, or Running a Macro.

As you write or edit your macros and procedures, you may make various mistakes as you create or alter the statements in your macros and procedures. VBA can detect many of these errors as you write the macro and detects other errors as you run the macro.

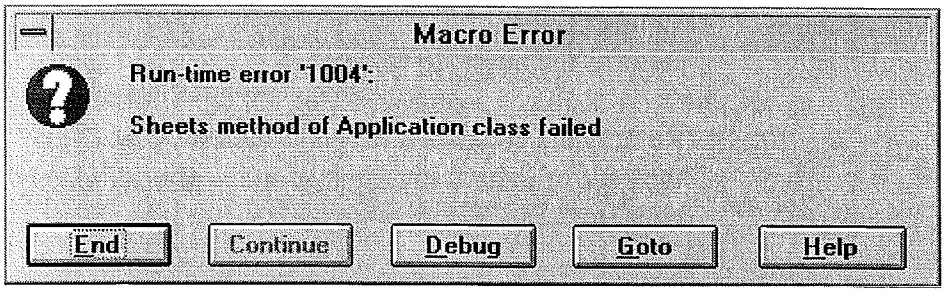
Syntax Errors

Syntax is the name given to the specific order of words and symbols that makes up a valid VBA statement.

Tip: If you need help resolving syntax error with a specific VBA keyword or built-in procedure (like `MsgBox`), position the insertion point over the keyword or procedure name, and then **press F1**. VBA displays the on-line help for that keyword or procedure (if there is any).

Runtime Errors

The error dialog box for runtime errors contains several command buttons. The following list summarizes each of these buttons:



1. End–Choose this command button to end the procedure.

2. Continue–Choose this command button to continue the procedure. Some runtime error allow you to continue running the procedure; for most runtime errors, however, this command button is disabled.

3. Debug–Choose this command button to use the VBA Debugger to help track down and solve the causes of whatever problem caused the runtime error.

4. Goto–Choose this command button to go to the line in the procedure source code that produced the runtime error. VBA displays the module sheet that contains the procedure that produced the error, positions the text in the module so that the offending line is displayed, and selects that line.

5. Help–This command accesses the VBA on-line help system and display the help topic describing the precise runtime error that has occurred. Use this button to get more information if it is not clear to you what the runtime error message means.

If you don't understand why the use of a particular VBA keyword or procedure causes a runtime error, you can get help with that specific keyword or procedure by positioning the insertion point over that word and **pressing F1**. VBA displays the online help topic for that keyword or procedure, if any.

4. Printing Your Macros

At some point in time, you'll probably want to print some of your macros. You might want to print a macro for archival or documentation purposes, to show to a colleague, or to study. (Studying the macros produced by the Excel Macro Recorder is a good way to help yourself earn VBA).

To printing a module sheet, follow these steps:

1. Select the module sheet or sheets you want to print. (You can select several module sheets at once by holding down the Ctrl Key and clicking on the sheets' name tabs to select them.)

2. Choose the **File** | **Print** command. Excel displays the Print dialog box.

3. To print only the selected module sheets, select the Selected Sheet(s) option in the Print What area of the Print dialog box. Leave this option unselected if you want to print the entire workbook (including worksheets as well as module sheets).

4. Fill in the other options in the Print dialog box as you would for any other printing job.

5. Choose OK. Excel prints the selected module sheets, or the entire workbooks, depending on whether you choose the Selected Sheet(s) option in Step 3.

5. viewing and Inserting Visual Basic's Functions

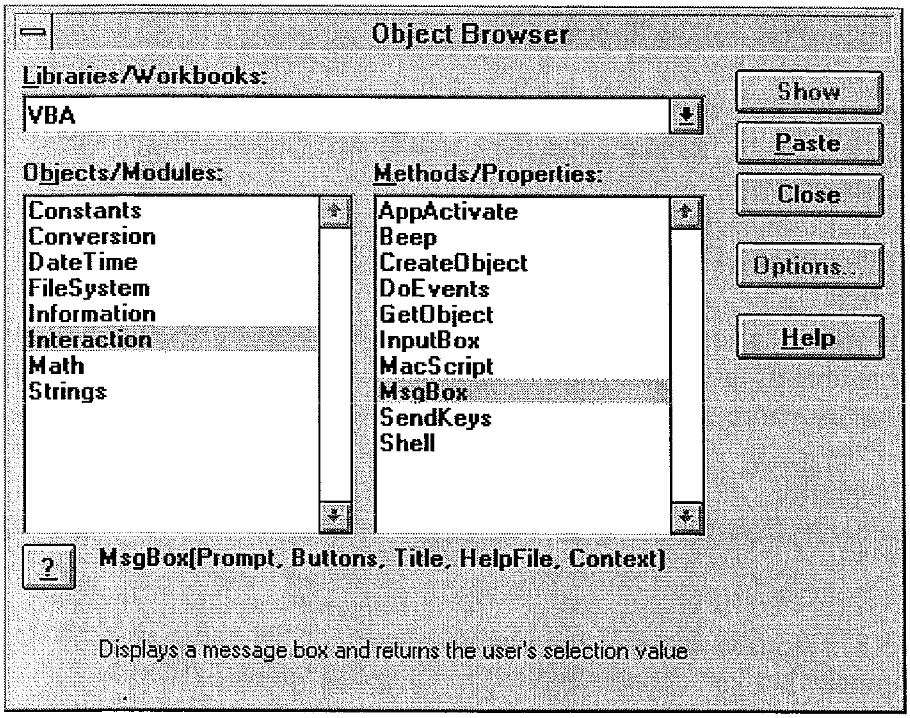
To insert a function call into your VBA source code, your start the Object Browser the same way you have already learned: choose the **View** | **O**bject Browser command-which appear on the **V**iew menu only when the current sheet is a module- or click the Object Browser button on the Visual Basic toolbar, if the toolbar is displayed.

Whichever technique you use to star the Object Browser, VBA displays the Object Browser dialog box shown in the next figure

1. Select **VBA** in the **L**ibraries/Workbooks drop-down list box at the top of the Object Browser dialog. The O**b**jects/Modules list now shows the various categories of functions, procedures, and constants defined by VBA.

2. Select the function category in which you are interested (Constants, Conversion, DateTime, FileSystem, Information, Interaction, Math, or Strings) in the Objects/Modules list. The figure shows the **Interaction** category selected.

3. Select the specific function you want to use-or want to get more information about- in the Methods/Properties list. Figure shown the MsgBox function selected.



At the bottom of the Object Browser dialog box in Figure, notice that the MsgBox function name and complete argument list appear, along with a simple explanation of the function's action and return value. Each name that appears in the argument list is the name to use for thc function's named arguments.

6. Viewing and Inserting Excel's Functions

The use the Object Browser to view the functions that Excel makes available to VBA, or to paste an Excel function into your source code, follow these steps:

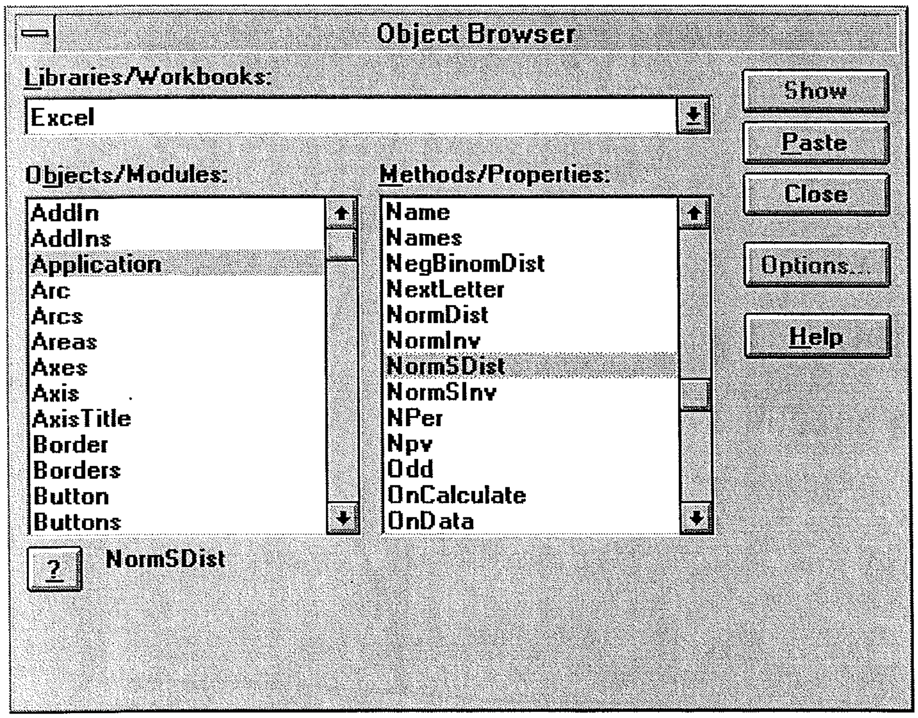
1. Open the Object Browser dialog box.

2. Select **Excel** in the Libraries/Workbooks drop-down list box at the top of the Object Browser dialog box. The Objects/Modules list now shows the various

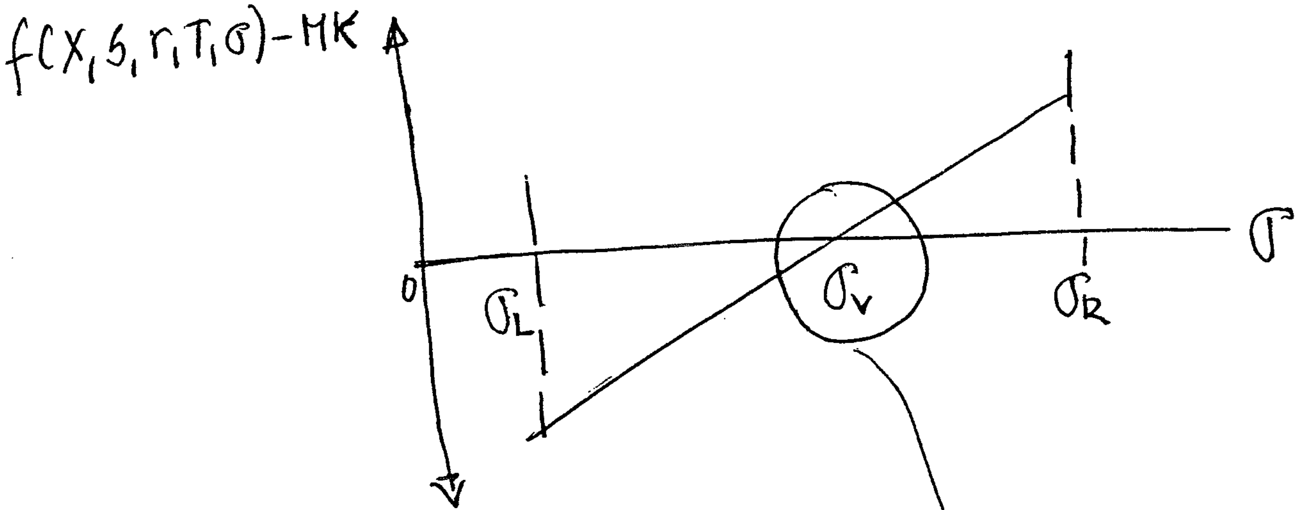
categories of functions, procedures, constants, and other program objects defined by Excel.

3. Select the **Application** category in the Objects/Modules list.(See next figure)

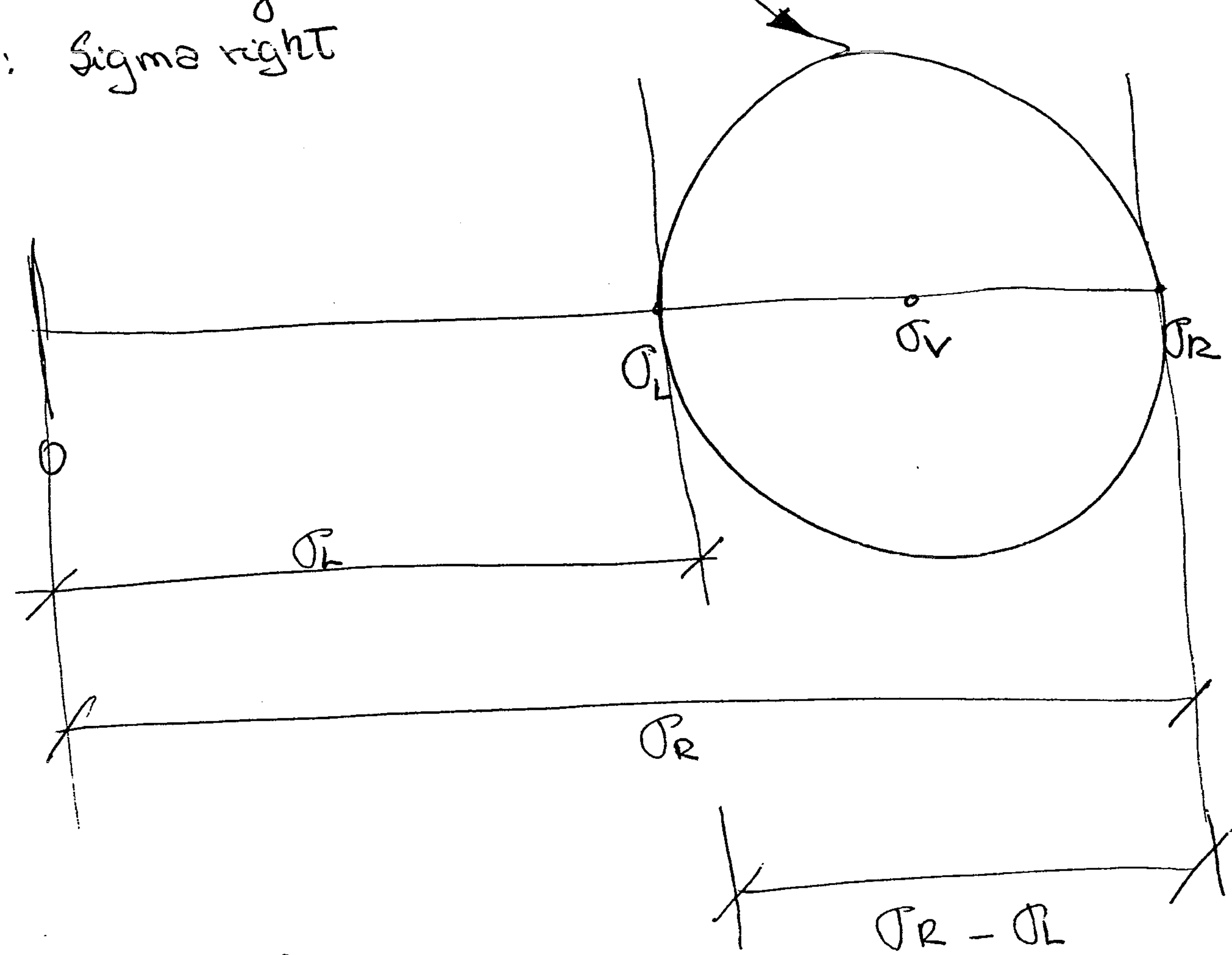
4. Select the specific Excel function you want to use-or want to get more information about-in the **M**ethod/Properties list. The following Figure shows the Excel **Normsdist** function selected.



Bisection Method



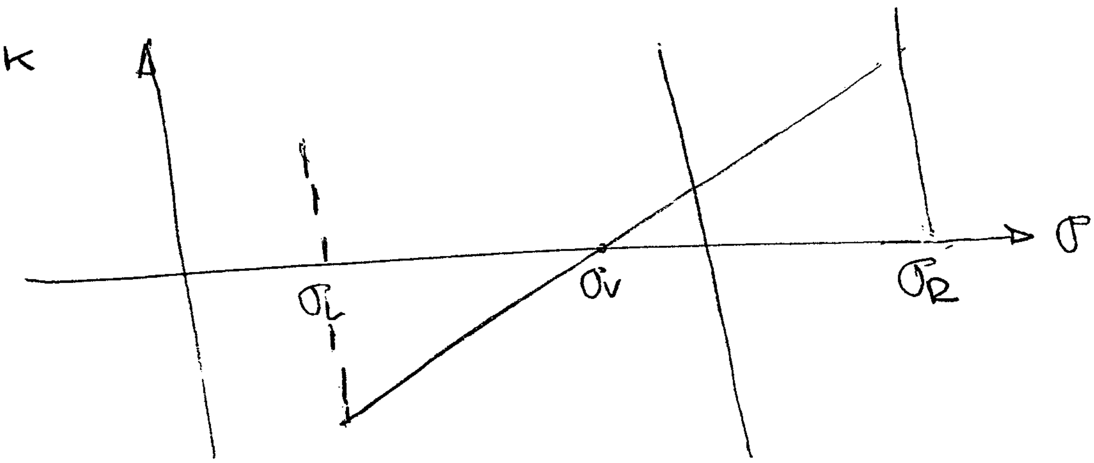
σ_L: Sigma left
σ_V: Volatility
σ_R: Sigma right



Do While $\frac{\sigma_R - \sigma_L}{2}$ > accuracy

Case 1

$f(X, S, r, T, \sigma) - MK$



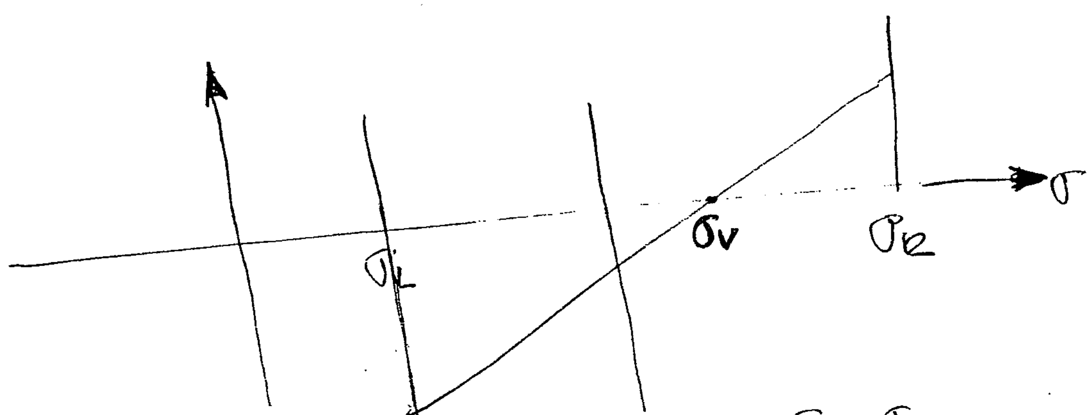
$\sigma_{Midpt} = \dfrac{\sigma_R + \sigma_L}{2}$

$fL = f(X, S, r, T, \sigma_L) - MK$ is negative

$fMdpt = f(X, S, r, T, \sigma_{Midpt}) - MK$ is positive

$fL * fMdpt$ is negative then $\sigma_R = \sigma_{midpt}$

Case 2



$\sigma_{midpt} = \dfrac{\sigma_R + \sigma_L}{2}$

$fL = f(X, S, r, T, \sigma_L) - MK$ is (-)

$fMdpt = f(X, S, r, T, \sigma_{Midpt}) - MK$ is negative

$fL * fMdpt$ is positive then $\sigma_L = \sigma_{midpt}$

Financial Data Lab

Assignment 5

Sofia Morote

INDEX

Financial Data Lab
Assignment 5

Sofia Morote

Swaps

Vanilla Swap

Create a swap price calculator (use Excel with VBA) which takes as an input the Euro$ futures prices curve up to 3 years in maturity, and for a swap with notional principal equal to $100m, produces the swap curve.

Compare your curve to that found on pages of the major swap market maker such as a Harlow Butler, Tradition or Pebron Yamane.

SWAPS

What is an interest rate swap?

An interest rate swap is an agreement between two parties. Each contracts to make payments to the other on particular dates in the future. One, known as the fixed rate payer, will make so-called fixed payments. These are predetermined at the outset of the swap. The other, known as the floating rate payer will make payments, the size of which will depend on the future course of interest rates.

The most common type of swap is a "plain vanilla[1]" interest rate swap. In this, one party, B, agrees to pay to the other party, A, cash flows equal to interest at a predetermined fixed rate on a notional principal for a number of years. At the same time, party A agrees to pay party B cash flows equal to interest at a floating rate on the same notional principal for the same period of time. The currencies of the two sets of interest cash flows are the same. The life of the swap can range from 2 year to over 15 years.

Features of a standard interest rate swap[2].

- **The notional principal**: Fixed and floating payments are calculated as if they were payments of interest on an amount of money borrowed or lent. This amount is referred to as the notional principal.

- **The fixed rate**: This is the rate applied to the notional principal to calculate the fixed amounts.

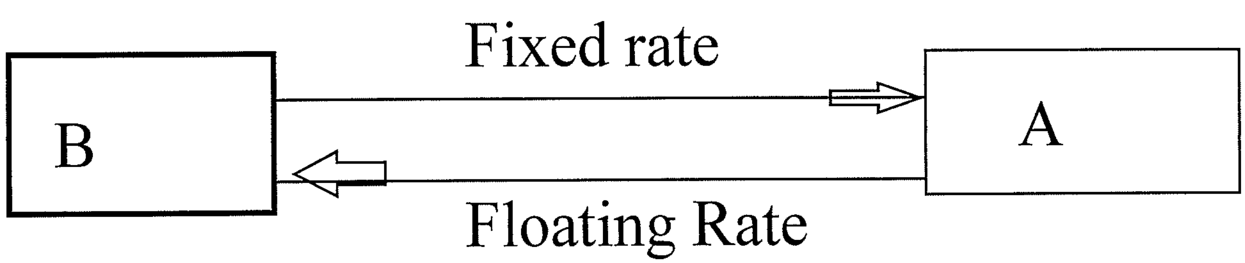
- **Dates of payment**: Fixed rate payments are usually paid either annually or every six months. For example, they might be paid every first day of February and August from 1 February 1991 until 1 August 1996. This last payment date is known as the termination date or, more commonly, the maturity date. Two other relevant dates are the trade date, on which the parties agree to do the swap and the effective date, when the first fixed and floating payments start to accrue. Note that, in general, no payments take place on either the trade date or the effective date.

- **The fixed rate payments**: Collectively, the fixed rate payments on a swap are known as the fixed leg. Each fixed payment is determined by the notional

[1] Options Futures, and Other Derivative securities by John Hull
[2] Swaps by Paul Miron and Philip Swannell.

Swaps

Vanilla Swap



Same:
Notional Principal
Number of years
currency

principal, by the fixed rate and by a quantity known as "the fixed rate day count fraction", according to:

$$\text{Fixed Amount} = \text{Notional Principal} \times \text{Fixed rate} \times \text{Fixed Rate Day Count Fraction}$$

Broadly, the "fixed rate day count fraction" will be equal to the fraction of a year since the previous payment (or since the effective date).

Suppose a swap has notional principal P and fixed rate R. A fixed rate payment is due on $D_2 = (d_2, m_2, y_2)$[3]. The prior fixed rate payment was on $D_1 = (d_1, m_1, y_1)$. In the United States "Actual/365(Fixed)"[4] is known as bond basis. When the fixed rate is paid on this basis:

$$\text{Fixed Rate Day Count Fraction} = \frac{D_2 - D_1}{365}$$

In the United States "Actual/360" is known as money market basis[5]. In this case:

$$\text{Fixed Rate Day Count Fraction} = \frac{D_2 - D_1}{360}$$

Lastly, if the payments are on an equal coupon basis then:

Fixed Rate		1	if fixed payments are annual
Day Count	=	1/2	if fixed payments are semi-annual
Fraction		1/4	if fixed payments are quarterly

3 For example, is D_2 were 17 May 1991 then $D_2 = (17,5,1990)$

4 "Fixed" here refers to the fact that 365 is ued regardless of leap years.

5 Take care. In the United Kingdom, money market basis means "Actual/365(fixed)".

PRICING A VANILLA SWAP

Input:

Dates: $t_1,...,t_n$ Forward/Futures rates: $f(t_1), f(t_2),...,f(t_{n-1,n})$

Output:

Swap Coupon

Consider
Daycounts: di = #days between ti-1 and ti.

STEP 1
Compute all the Forward Rates or get them from Euro$ futures prices

LIBOR gives us: dates $t_1,...,t_n$ and deposit rates $r(t_1),...,r(t_n)$.
Here, $r(t_i)$ is the deposit rate if you deposited $ today and received the money at time t_1.
The forward rates are calculated as follows.
a) f(t1) = r(t1)
b) $f(t_{12})$ solves: $(1+f(t_1)d_1/360)(1+f(t_{12})d_2/360) = (1+r(t_2)(d_1 + d_2)/360)$
c) etc.

```
Euro$ futures tell you the f_ij directly
The ric code is ED: <f3>
the value of f_ij = 100 - Euro$ futures
```

STEP 2

Find the final value of $1 rolled over at these forward/future rates:

$V = (1+f(t_1)d_1/360)*(1+f(t_{12})d_2/360) *...*(1+f(t_{n-1,n})d_n/360)$

then, $1 now "becomes" $V at time t_n, which is $D = \Sigma d_i$ days from now.

STEP 3

Find the effective annual yield based on a 360-day

The effective annual yield based on a 360-day year is the interest rate, r, such that

$$V = 1 * (1 + r)^{(D / 360)}.$$

Here, r is the annually compounded interest rate based on a 360 days year.

Why is this the effective annual yield: Suppose I invested \$1 for 1 year and at the end of the year I got \$F. The effective annual yield is the interest rate, compounded annually, which makes \$1 today equal to \$F in one year, i.e. the r that solves.

$$F = \$1 * (1 + r).$$

If I invest it for 1 day and get G at the end of 1 day, the effective annual yield is the r that solves

$$G = \$1 \ (1 + r)^{(1 / 360)} \quad \text{based on a 360-day year.}$$

The r we have calculated is then:

$$r = V^{(360 / D)} - 1$$

STEP 4

Convert this r into a 365-day year

The conversion here is a little strange in that it is done on the basis of continuous compounding. Basically, we solve for r' such that

$$\exp(r(n/360)) = \exp(r'(n/365))$$

The first expression is the effect of continuous compounding over n days based on a 360-day year. The second is the same on a 365-day year.

Therefore, $r' = r(365/360)$.

Why is this a little strange: because we could have directly computed the effective annual yield on the basis of a 365-day year, i.e. computed $r' = V^{(365 / D)} - 1$

What do we use r' for: we use it to discount all cash flows on the fixed side.

STEP 5

The fifth step is to use a simple fact from the annuity/bond valuation formula. Suppose we have a coupon bond, the coupon is paid quarterly, the coupon rate is c and the yield to maturity is r. Then, the price of the bond is:

$$P = \frac{cF/4}{(1+r/4)} + \frac{cF/4}{(1+r/4)^2} + \frac{cF/4}{(1+r/4)^3} + \ldots\ldots + \frac{cF/4 + F}{(1+r/4)^n}$$

when there are n-quarters to go.
F: Notional Principal = \$100m

The bond sells at par (i.e. P = F) when c = r.
Therefore, to determine the **swap coupon**, we need to set c = r'.

Note: if you set the swap coupon this way, the value of the fixed side will be close to par, but not quite. This is because the coupon is calculated assuming a 360-day year, but actual payments are made on reset dates. To get an exact answer, you have to use **an iterative method using the actual daycounts.Use Excel programming VBA for find swap coupon**

STEP 6

Compare your curve to that on pages of major swap market maker such as Harlow Butler or Tradition or Pebrom Yamane.
You can find that pages in PTW:
SWAP <enter>
SWAP / 1 <enter>

EXCEL PROGRAMMING WITH VBA[1]

Understanding Single-Dimensional Arrays

element 0	10.3
	23
	4
	12.3
element 4	6

The array above has five elements in it; each element stores a Double type number. Notice that the elements in the array are numbered from 0 to 4, for a total of 5 elements.

For example, if the array is named NumArray, then the following statement assigns the number `12.3` to the variable AnyNum:

```
AnyNum = NumArray(3)
```

In this statement, the number 3 is the array subscript; notice that it is enclosed by parentheses, and is *not* separated with any spaces from the array's name. Because element numbering starts with 0, the element that this statement references is actually the 4th element of NumArray.

col 0	col 1
10.3	10.2
2.5	22
4	4.2
12.3	11
6	0.2

Two-dimensional Array

[1] Excel Programming with Visual Basic for Applications by Matthew Harris pages 471-475 Sams Publishing.

If the array in Figure above is named **NumTable**, then the following statement assigns the value `10.2` (from the first row in the 2nd column of the array) to the variable **AnyNum**:

```
Anynum = NumTable(1, 0)
```

Similarly, the following statement stores the value `2.5` in the second row of the 1st column of the array:

```
NumTable(0, 1) = 2.5
```

In both of the preceding statements, notice that the subscripts to the array are enclosed in parentheses, and that the column and row coordinates are separated by commas.

The *Option* Base Statement

The **Option Base** statement allows you to specify 0 or 1 as the default starting number for array subscripts. If you don't use the **Option Base** statement, then VBA starts array subscript numbering at 0 (the default). You must place the **Option Base** statement in the declaration area of a module, before any variable, constant, or procedure declarations. You can't place the **Option Base** statement inside a procedure.

The next two statement show examples of the Option Base compiler directive:

```
Option Base 0      |     the default setting
Option Base 1      |     array subscripts start with 1
```

Declaring Arrays

The general syntax for declaring an array with the **Dim** statement is:

```
Dim VarName([Subscripts]) [As Type]
```

`VarName` represents any name for the array that meets VBA's rules for identifier names. The *`Subscripts`* clause represents the dimension(s) of the array. You may declare arrays with up to 60 dimensions. For a single-dimensional array, include one *`Subscripts`* clause; for a two dimensional array, include two *`Subscripts`* clauses (separated by a comma), and so on, for as many dimensions as you wan your array to have. Each *`Subcripts`* clause adds a new dimension to the array.

The following examples are all valid array declarations:

```
Dim January(1 to 31) As String
Dim January(31) as String   'assumes Option Base 1
Dim LookupTable( 2, 10)    'assumes Option Base 1
Dim HexMultiplitacion(0 to 15, 0 to 15) As String
Dim LookupBook(1 To 3, 1 To 2, 1 To 10)
```

Using Arrays

The general syntax for accessing an array element is:

```
arrayName(validIndex1, [validIndex2]...)
```

`arrayName` represents the name of an array. `validIndex1` represents a valid subscript value for the firsts dimension of the array. `validIndex2` represents a valid subscript value for the second dimension of the array, if there is one. You must supply a subscript value for every dimension in the array, every time you access an element in the array.

The following code fragment shows a typical array declaration and usage:

```
Dim Factorial(0 To 30) As Double
Factorial(0) = 1
For I = 1 To 30
       Factorial(I) = I * Factorial(I - 1)
Next I
```

Examples next page

```
Option Base 1
Sub practice( )

Const Array_max As Integer = 8
Const Max As Integer = 8
Const Maxi As Integer = 8
Dim I As Integer
Dim Forward(Max) As Double
Dim hundred(Array_max) As Integer
Dim daycount(Maxi) As Integer

'creates an array of 100 and stores in hundred(I)

For I = 1 To Array_max
   hundred(I) = Int(100)
   Next I

'creates an array subtracting 100 - values of column 2 and displays in column 3
For I = 1 To Max
   Forward(I) = hundred(I) - Cells(I, 2).Value
   Cells(I, 3).Value = Forward(I)
   Next I

'creates an array for the number between dates of the first column and displays
' in column 4
For I = 2 To Maxi
   daycount(I) = Cells(I, 1) - Cells(I - 1, 1)
   Cells(I, 4).Value = daycount(I)
   Next I
End Sub
```

18-Nov-95	93.4	6.6000	
17-Feb-96	93.6	6.4000	91
18-May-96	93.6	6.4000	91
17-Aug-96	94.1	5.9000	91
16-Nov-96	94.3	5.7000	91
15-Feb-97	94.5	5.5000	91
17-May-97	94.7	5.3000	91
15-Aug-97	94.9	5.1000	90

```
'Example: Fragments of a program

Option Base 1

'maximum array elements
Const ARRAY_MAX As Integer = 15

'minimum number to be entered
Const ARRAY_MIN As Integer = 3

Sub DemoStaticArray()
  'declare single-dimensional array
  Dim NumArray(ARRAY_MAX) As Double
  Dim aSum As Double    'for sum of numbers
  Dim Count As Integer  'loop counter
  Dim cLow As Integer   'low limit for sum
  Dim cHigh As Integer  'high limit for sum
   Dim oldSheet As String 'original sheet name


    'preserve original sheet name
        oldSheet = ActiveWorkbook.ActiveSheet.Name

  'select a new sheet
        ActiveWorkbook.Sheet("Sheet1").Select


  'clear the worksheet cells for later display
  'of the array's contents
        For Count = 1 To (ARRAY_MAX + 2)
        Cells(Count, 1).Valuc = ""
        Cells(Count, 2).Value = ""
        Next Count
  ......................................
  'initialize the sum to 0, then loop through array
  'to compute the sum for the specified range
        aSum = 0
        For Count = cLow To cHigh
        aSum = aSum + NumArray(Count)
         Next Count
```

Financial Data lab

Sofia Morote

Assignments 6/7

- Description of the assignments.
- Using PTW to get data options on a future
- Black Model (For assignment 6)
- Put-call parity (For assignment 7)
- Excel with VBA

Financial Data Lab

Assignment 6

Sofia Morote

Forward Volatilities

In implementing interest rate models, we frequently need estimates of volatilities of future interest rates. One way to estimate these volatilities is to calculate the volatilities implicit in the prices of options on Eurodollar futures.

Use the Black model for options on futures to calculate the implied volatilities for option maturities up to 2 years from now at quarterly intervals. This will produce the calculations of "forward volatilities".

Financial Data Lab

Assignment 7 (optional)

Sofia Morote

Futures Options

There is an important relationship between put and call :

C+Xexp(-r*T) = P+Fexp(-r*T)

This relationship is know as put-call parity

Put-call parity shows that the value of a call with a certain exercise price and exercise date can be deducted from the value of put with the same exercise price and date, and vice versa.

Consider two portfolios:

Portofio A: There is one future call option plus an amount of cash equal to Xexp(-r*T)

Assignment 7 (optional)

Portfolio B: There is one future put option plus Fexp(-r*T)

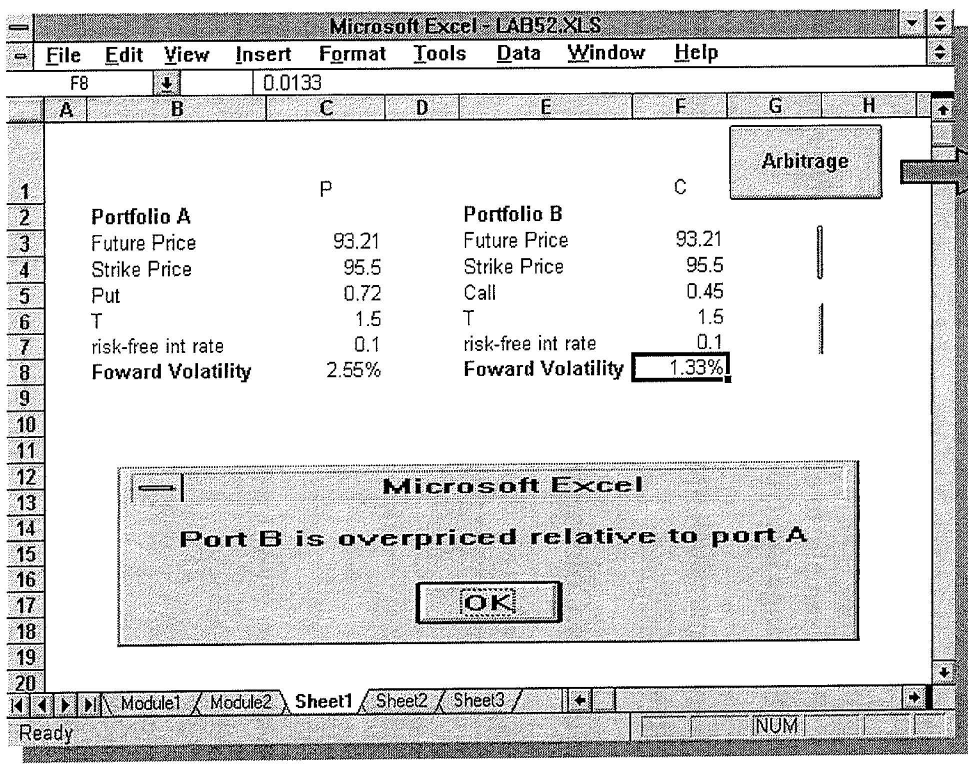
Use these values in the Eurodollar Future option Dec. 97.

Create a program.The program will acomplish the following:

Display a message informing if there is an arbitrage opportunity and recomending arbitrage strategy.

Based on the Black model for options on futures, calculate the implied volatility for the underlying option.

Example assignment 7



macro Button

Using PTW for get the data

To display an option on a future:

- Type the *RIC*
- Enter the Month Code.
  Note: futures month codes are used to chain options on futures.
- Enter the Year Indicator.
- Enter the Chain Identifier (+ for options on futures).
- Press the <F3> key
  For Example, to chain the Gold Option for April 1992 on the Commodity Exchange Inc. type: **GCJ2 + <F3>**

Instruments which use this procedure are described in the directory as Futures and Options.

Futures Contracts

Month	Code
January	F
February	G
March	H
April	J
May	K
June	M
July	N
August	Q
September	U
October	V
November	X
December	Z

For the Assignment you will get the data of options on Eurodollars futures, Dec 97
Then the Ric code will be **EDZ7+**
ED: Eurodollar Futures
Z : December
7: 1997

THEORY

WHAT IS AN OPTION?

A call (put) option is the contract right to buy (sell) specified amount of some real or financial asset at a fixed price on or before a given date.

If the option purchaser acts upon this right to buy, he or she is *exercising* this right; and the fixed price of the transaction is known as the *strike* price. The seller of the option, known as the *writer*, must be prepared to sell the specified asset when the option purchaser exercises these rights. When the option buyer exercises, the seller is *assigned*. The *maturity* of the contract is known as the *expiration* date, and exchange option trading takes place in any one of a number of set contract months, or *cycles*. An *American* option allows the holder to exercises the right any time before the expiration, and a *European* option restricts the right only to expiration and not before.

Black's Model for FUTURE OPTIONS

In a Future option (or options on future), the underlying asset is futures contract.
Black's Model shows that a futures price can be treated in the same way as a security paying a continuous dividend yield at rate *r*. The European call price, c, and European put price, p, for a futures option are therefore given by equations:

$$\mathbf{c = e^{-r(T-t)}\ [\mathit{F}\ \mathit{N}(d_1)\ -\ \mathit{X}\ \mathit{N}(d_2)]}$$

$$\mathbf{p = e^{-r(T-t)}\ [\mathit{X}\ \mathit{N}(-d_2)\ -\ \mathit{F}\ \mathit{N}(-d_1)]}$$

where

$$d_1 = \frac{\ln(F/X) + (\sigma^2/2)(T - t)}{\sigma\ (T - t)^{\wedge}0.5}$$

$$d_2 = \frac{\ln(F/X) - (\sigma^2/2)(T - t)}{\sigma\ (T - t)^{\wedge}0.5} = d_1 - \sigma\,(T - t)^{\wedge}0.5$$

There are a reasonable approximation for futures contracts on stocks, stock indices, and currencies. They are also reasonable for most commodity futures. However, they are questionable when the asset underlying the futures contract is an interest-rate-dependent security such as a Treasury bond or Treasury bill.

PUT-CALL PARITY

A put-call parity relationship for European futures options can be derived. If F_T is the futures price at maturity, a European call plus an amount of cash equal to $Xe^{-r(T-t)}$ has the terminal value

$$\max(F_T - X,\ 0) + X = \max(F_T, X)$$

An amount of cash equal to $Fe^{-r(T-t)}$ plus a futures contract plus a European put option has terminal value.

$$F + (F_T - F) + \max(X - F_T, 0) = \max(F_T, X)$$

Now, we can derive an important relationship between p and c. Consider the following portfolios:

Portfolio A: One European future call plus an amount of cash equal to $\boldsymbol{X}\ e^{-r(T-t)}$
Portfolio B: One European future put options plus an amount of cash equal to $\boldsymbol{F}\ e^{-r(T-t)}$

Since the two portfolios are equivalent at maturity, it follows that they are worth the same today. The futures contract is worth zero today. Hence

$$c + X\ e^{-r(T-t)} = p + F\ e^{-r(T-t)}$$

This relationship is known as put-call parity. It shows that the value of a European call with a certain exercise price and exercise date can be deduced from the value of a European put with the same exercise price and date, and vice verse..

If Equation Put-call parity does not hold, there are arbitrage opportunities. Suppose that the Future price is $31, the exercise price is $30, the risk-free interest rate is 10 percent per annum, the price as a 3-month European call option is $3, and the price of a 3-month European put option is $2.25. In this case,

$$c + Xe^{-r(T-t)} = 3 + 30e^{-0.1x0.25} = 32.26$$

$$p + F\,e^{-r(T-t)} = 2.25 + 30.23 = 32.5$$

Portfolio B is overpriced relative to portfolio A.

The correct arbitrage strategy is to buy the securities in portfolio A and short the securities in portfolio B.

For alternative situation, suppose that the call price is $3 and the put price is $1. In this case

$$c + Xe^{-r(T-t)} = 3 + 30e^{-0.1x0.25} = 32.26$$

$$p + F\,e^{-r(T-t)} = 1 + 30.23 = 31.23$$

Portfolio A is overpriced relative to portfolio B.

The correct arbitrage strategy is to short the securities in portfolio A and buy the securities in portfolio B.

EXCEL PROGRAMMING WITH VISUAL BASIC FOR APPLICATIONS (VBA)

Using the Select....Case Statement

The example of nested If..Then...Else (pag 8, last hand-out) easily make a two-way decision, but what if you need to choose between five, eight, ten differents courses of action?

Fortunately, VBA offers a conditional braching statement for use when you must choose among a large number of different branches: The `Select Case` Statement.

The Select Case statement has the following general syntax:

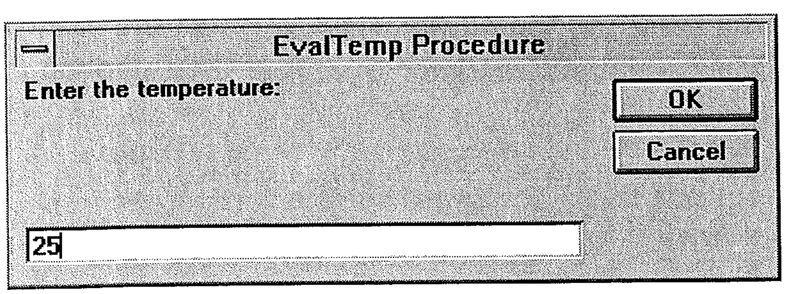
```
Select Case TestExpression
          Case ExpressionList1
               statement1
          Case ExpressionList2
               statement2
            .
            .
            .
            Case ExpressionListN
                statementN
              [Case Else
                          ElseStatements]
End Select
```

TestExpression is any numeric or string expression. *ExpressionList1, ExpressionList2, and ExpressionListN* each represent a list of logical expressions, separated by commas. *Statements1, statements2, statementsN, and ElseStatements* each represent none, one, or several VBA statements. You can include as few or as many `Case ExpressionList` clauses in a Select Case statement as you which.

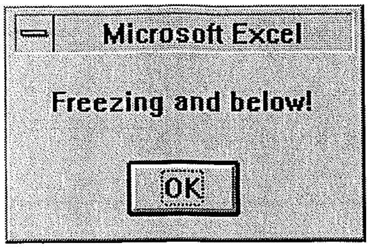
'Listing 5 Example Select case

```
Sub EvalTemperature()
  Dim temperature

 temperature = Application.InputBox( _
                 prompt:="Enter the temperature:", _
                 Title:="EvalTemp Procedure", _
                 Type:=1)

   Select Case temperature
    Case Is > 100
     MsgBox "Too hot!"
    Case 75 To 100
     MsgBox "Stay cool!"
    Case 50 To 74
     MsgBox "Okay."
    Case Is > 32
     MsgBox "Pretty cold."
    Case Else
     MsgBox "Freezing and below!"
  End Select
End Sub
```

When you run the program EvalTemperature, a Msgbox appear



Then you will obtain a comment about the temperature that you just entered:



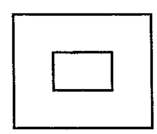
ASSIGNING A MACRO TO A BUTTON ON A SHEET.

In Microsoft Excel, you can create a button on a worksheet or chart sheet and then assign a macro to it. By attaching a macro to a button, you make it visible and readily, available while you are working. If the button appears on a worksheet, for example, the macro is available every time you open that worksheet.

⇒ **To create a button on a sheet and assign a macro to it.**

Before you do this procedure, you must have the Drawing toolbar display. Use the Toolbars command on the View menu to display the toolbar.

1. Click the Create Button on the Drawing toolbar.



2. Point to where you want one corner of the button.

3. Drag until the button is the size and shape you want. (When you release the mouse button, the Assign Macro dialog box appears.

4. To assign an existing macro to the button, type or select the name of the macro in the Macro Name/Reference box, and then choose the OK button.