REFERENCES

Thai

เครก สตินสัน. คู่มือการใช้งาน Windows 3.1. กรุงเทพมหานคร: บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2536.

ดวงแก้ว สวามิภักดิ์. การโปรแกรมภาษา C. กรุงเทพมหานคร: บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2531.

นที ชาญศิลปากร. CAI: ห้องสมุดบนพีซีมาแล้วจ้า. Computer Time 1 (สิงหาคม - กันยายน 2536): 16-17.

มนตรี พจนารถลาวัณย์. การเขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบซี. กรุงเทพมหานคร: บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2537.

เอกสารประกอบการเรียนวิชา Unit Operation I (laboratory). กรุงเทพมหานคร: ภาควิชาวิศวกรรมเคมี คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2536.

English

Bird, R.B., Stewart, W.E., and Lightfoot, E.N. Transport Phenomena. Singapore: John Wiley & Sons, 1960.

Bitter, G.G. Microcomputers in Education Today. New York: Mitchell Publishing, 1989.

Boardman, A.D., Cooper, G.S., James, B.W., Keeler, G.J., and Seage, J. Software Development for Undergraduates in Physics. Comput. Educ. 12 (1988): 29-35.

Boggan, S., Rarkas, D., and Welinske, J. Developing Online Help for Windows. Indiana: Sam Publishing, 1993.

Borland International, Inc. ObjectWindows for C++: User's Guide. California: Borland International, 1991.

______. Resource Workshop: User's Guide. California: Borland International, 1991.

______. Turbo C++ for Windows Version 3.0: Programmer's Guide. California: Borland International, 1991.

Brown, G.G., Katz, D., Foust, A.S., and Schneidewird, R. Unit Operations. New York: John Wiley & Sons, 1950.

Chopey, N.P. Handbook of Chemical Engineering Calculations. 2nd ed. Singapore: McGraw-Hill, 1994.

Coad, P., and Yourdon, E. Object-Oriented Analysis. 2nd ed. New Jersey: Prentice-Hall, 1991.

Coulson, J.M., and Richardson, J.F. Chemical Engineering: Volume 1. 3rd ed. Oxford: Pargamon Press, 1977.

Coulson, J.M., Richardson, J.F., Backhurst, J.R., and Harker, J.H. Chemical Engineering: Volume 2. 3rd ed. (SI Unit). London: A. Wheaton Co. Ltd., 1978.

Foust, A.S., Wenzel, L.A., Clump, C.W., Maus, L., and Andersen, L.B. Principles of Unit Operations. 2nd ed. Singapore: John Wiley & Sons, 1980.

Geankoplis, C.J. Transport Processes and Unit Operations. 2nd ed. New York: Allyn and Bacon, 1983.

Harrison, A.J.L., Banwell, J.K., Frost, M., and Plumbridge, W.J. An Interactive Teaching Experiment in Materials Science. Comput. Educ. 12 (1988): 119-124.

Holland, F.A., and Chapman, F.S. Liquid Mixing and Processing in Stirred Tanks. New York: Reinhold Publishing Coporation, 1966.

Kunii, D., and Levenspiel, O. Fluidization Engineering. New York: John Wiley & Sons, 1969.

Mathews, J.H. Numerical Methods for Mathematics, Science, and Engineering. 2nd ed. New Jersey: Prentice-Hall, 1992.

McCabe, W.L., Smith, J.C., and Harriott, P. Unit Operations of Chemical Engineering. 5th ed. Singapore: McGraw-Hill, 1993.

Murray III, W.H., and Pappas, C.H. Windows Programming: An Introduction. California: Osborn McGraw-Hill, 1990.

Perry, J.H. Chemical Engineering Handbook. 3rd ed. New York: McGraw-Hill, 1950.

Pfaffenberger, B. Que's Computer User's Dictionary. 3rd ed. New York: Lloyd J. Short, 1992.

Porter, A. C++ Programming for Windows. California: Osborn McGraw-Hill, 1993.

Raston, A., and Reilly, E.D. Encyclopedia of Computer Science. 3rd ed. New York: Chapman & Hall, 1993.

Rushton, J.H., Costich, E.W., and Everett, H.J. Power Characteristics of Mixing Impellers Part I. Chem. Eng. Progr. 46 (August 1950): 359-404.

______. Power Characteristics of Mixing Impellers Part II. Chem. Eng. Progr. 46 (September 1950): 467-476.

Scanlon, E., and Smith, R.B. A Rational Reconstruction of a Bubble Chamber Simulation Using the Alternate Reality Kit. Comput. Educ. 12 (1988): 199-207.

Schildt, H. Turbo C++ for Windows: Inside & Out. California: Osborn McGraw-Hill, 1992.

Shammas, N.C. Teach Yourself Turbo C++ Visual Edition for Windows in 21 Days. Indiana: Sam Publishing, 1994.

Smith, J.T. C++ Applications Guide. New York: McGraw-Hill, 1992.

Yusof, K.M., Sin, Y.F., Seong, T.B., Yunus, R., Ripin, A., and Aziz, B.A. Multimedia in Chemical Engineering Education and Training. The 1995 Regional Symposium on Chemical Engineering. Bangkok: Chulalongkorn University Press, 1995.

APPENDIX A

CREATING SIMPLE WINDOWS APPLICATION WITH TURBO C++ FOR WINDOWS

A-1. The Components of a Window

Microsoft Windows is a graphics-oriented and multitasking operation system providing several hundred API (Application Program Interface) functions. It basically provides many graphic user interfaces (GUI).

The components of a standard window is illustrated in figure A-1.1.

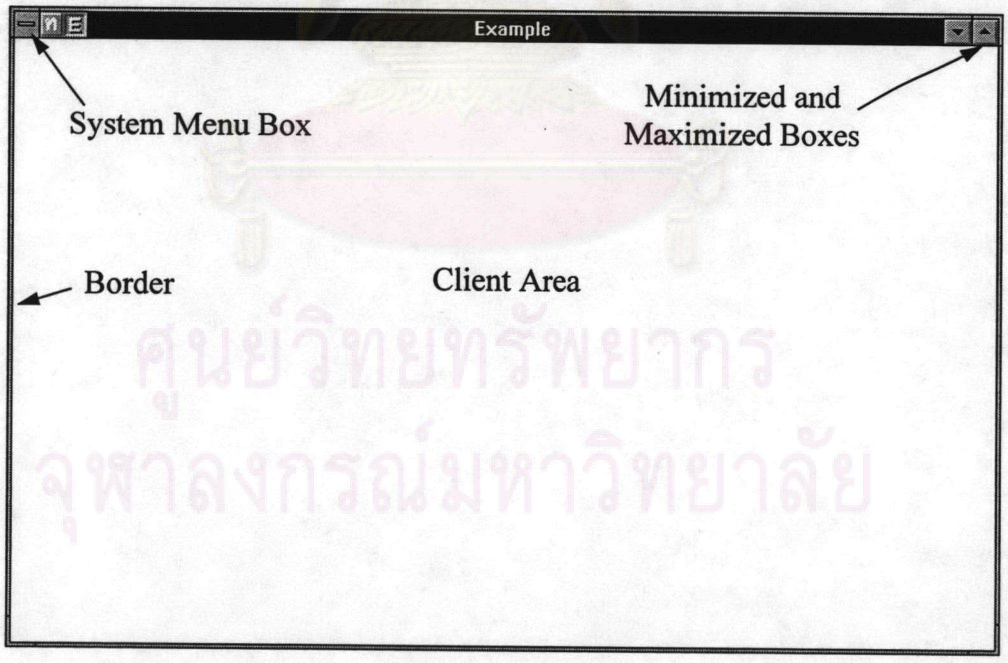


Figure A-1.1 The components of a standard window.

A border is the limits of the window and used for moving or resizing the window. At the top of the window is a title bar. System menu box is on the far left.

It will display the system menu when the user clicks on this box. At the far right are the minimize and maximize boxes. The program activity takes place in the client area. Horizontal and vertical scroll bars that are used to move text through the window also appear in most windows (it is not shown in figure).

A-2. Creating Resource files

Resources are data that define the visible portion of the Windows program. Resource Workshop is a tool that integrates the entire process of designing and compiling resources for applications running under Microsoft Windows, Version 3.0 and later. The programmer can modify the visual interface of Windows application easily by using this tool.

Resource Workshop supports the following types of Windows resources.

1. Dialog box : It is a pop-up window that communicates information to the user and lets the user select choices. It usually includes controls, such as edit boxes, combo boxes, radio buttons, and so on.
2. Menu : Menu bar is usually included in Windows application to show the lists of user choices.
3. Accelerator : It is a hotkey that the user presses to perform a task in an application. In general, it is used to duplicate menu functions, allowing the user to choose the function either by selecting from a menu or by pressing accelerator keys.
4. String table : It contains text that is displayed as a part of a Windows program.
5. Bitmap : It is a binary presentation of a graphic image in a program.
6. Icon : It is a small bitmap that used to represent minimized windows.
7. Cursor : It is a small bitmap that represents the position of the mouse on the screen.
8. Fonts.
9. User-defined and rcdata resources.

After creation all required resources, the programmer must save all resource code in .RC format, and then create the identifier file that defines numbers to uniquely identify each resource. In C language users can use "#define" and store them in a header file (.H). Coding 1 is an example for a typical header file.

Coding 1 The source code of a header file.

```
//example.h
#define MATERIAL          500
#define ID_DENUNIT        161
#define ID_DENSITY        162
#define ID_SPHERICITY     164
#define ID_SHAPE          165
#define CM_CONDITION      109
#define CM_MATERIAL       110
#define CM_EXPDATA        111
#define CM_SIMULATE       112
#define CM_RESULT         113
#define CM_GRAPH          114
#define CM_CONTENT        115
#define CM_SEARCH         116
```

A-3. Creating Program Codes

A-3.1 Minimal ObjectWindows Application

Coding 2 consists of source code to display the string "Example" on the title bar as shown in figure A-1.1. It begins with declaring a descendant of class Tapplication. The WinMain function is used for starting Windows application.

Coding 2 The source code for minimal Windows application.

```
#include <owl.h>

//Define an application.
class AppName : public TApplication
{
public:
  AppName(LPSTR AName, HINSTANCE hInstance,
     HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow):
         TApplication(AName, hInstance, hPrevInstance,
         lpCmdLine, nCmdShow){};
protected:
  virtual void InitMainWindow();
};

void AppName::InitMainWindow()
{
  MainWindow = new AppWindow(NULL, Name);
}

//———— Entry point of windows program————
int PASCAL WinMain(HANDLE ThisInstance, HANDLE PrevInstance,
                LPSTR Args, int VidMode)
{
 AppName App("Example",ThisInstance,PrevInstance,Args,VidMode);
 App.Run();
 return App.Status;
}
```

A-3.2 Adding a Menu

Programmers must create menu resource and the header file before add the menu in an application. An accelerator key can also be created in this step. Programmer must add the "AssignMenu("Menu Name")" statement to the Twindow deriving class constructor. Programmer can add functions in the Twindow deriving class to receive message from the user when select menu item. Source code for this part is shown in coding 3.

Coding 3 Adding menu and menu selection command in your application.

```
class AppWindow : public TWindow
{
 public:
  AppWindow(PTWindowsObject AParent, LPSTR Atitle);
  virtual void CMMaterial(RTMessageMsg)=[CM_FIRST+CM_MATERIAL];
  virtual void CMCondition(RTMessage Msg)=[CM_FIRST+CM_CONDITION];
              ...
}

AppWindow::AppWindow(PTWindowsObject AParent, LPSTR ATitle):
        TWindow(AParent, ATitle)
{
 AssignMenu("MYMENU");
}

void AppWindow::CMMaterial(RTMessage)
{
        ...
}
```

A-3.3 Control Dialog Boxes

The dialog box is usually called from a menu to receive data which is input by the user. The command for calling the dialog box is

```
GetApplication()->ExecDialog(new dialog class(this,dialog name));
```

For example, to call the MATERIAL dialog box in MyDialog1 class when receive CM_MATERIAL message:

```
void AppWindow::CMMaterial(RTMessage)
{
  GetApplication()->ExecDialog(new MyDialog1(this,MATERIAL));
}
```

One dialog box can have many controlling tools, such as edit boxes, combo boxes, and so on. Figure A-3.1 is an example for writing dialog box controlling code by using Resource Workshop.

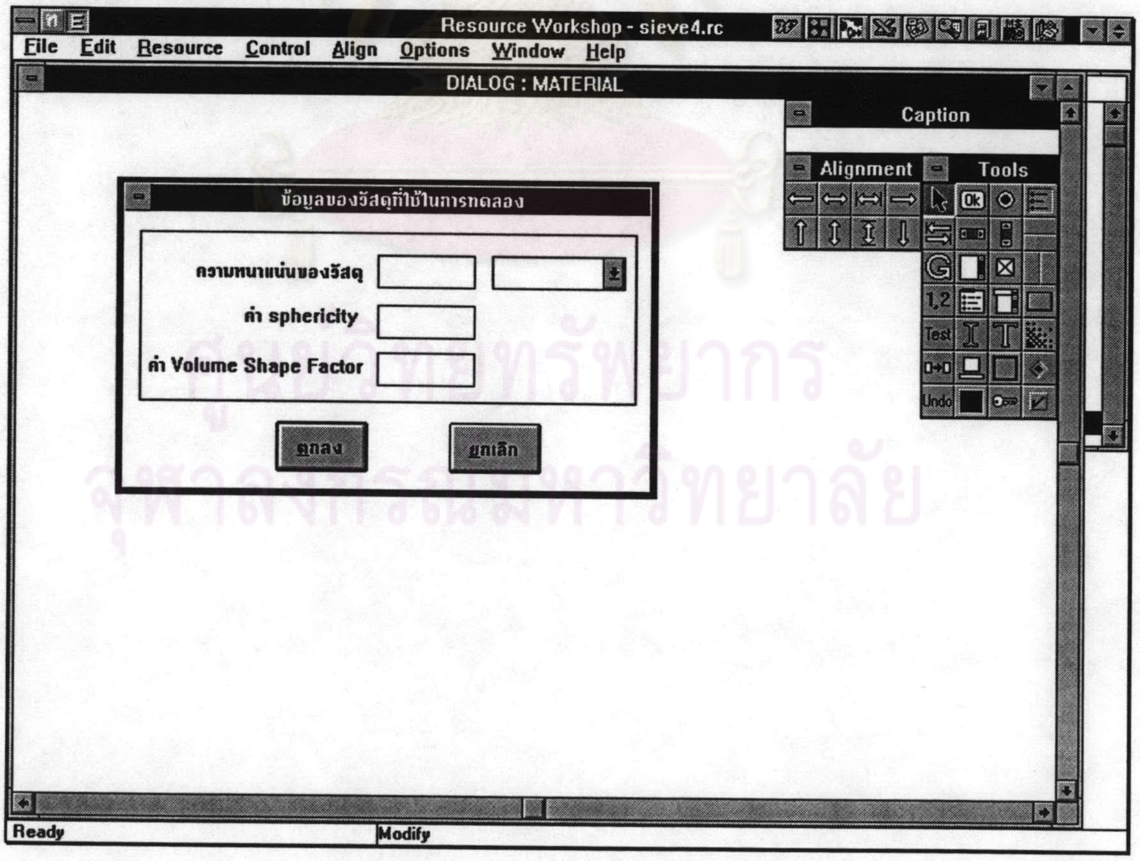


Figure A-3.1 Dialog box created in Resource Workshop.

This dialog box is a modal dialog box with edit boxes, combo box, push buttons, and text. The source code for this dialog resource is shown in coding 4.

Coding 4 Source code for dialog that shown in figure A-3.1.

```
MATERIAL DIALOG 38, 42, 190, 99
STYLE DS_MODALFRAME|WS_POPUP|WS_CAPTION|WS_SYSMENU
CAPTION "ข้อมูลของวัสดุที่ใช้ในการทดลอง"
FONT 12, "felv"
BEGIN
        CONTROL "", ID_DENSITY, "EDIT", ES_RIGHT | WS_CHILD |
        WS_VISIBLE | WS_BORDER | WS_TABSTOP, 91, 15, 35, 12
        CONTROL "", ID_DENUNIT, "COMBOBOX", CBS_DROPDOWNLIST |
                CBS_SORT | WS_CHILD | WS_VISIBLE | WS_VSCROLL |
                WS_TABSTOP, 132, 15, 49, 33
        EDITTEXT ID_SPHERICITY, 91, 33, 35, 12, ES_LEFT | WS_CHILD |
                WS_VISIBLE | WS_BORDER | WS_TABSTOP
        CONTROL "", ID_SHAPE, "EDIT", ES_RIGHT | WS_CHILD |
                WS_VISIBLE | WS_BORDER | WS_TABSTOP, 91, 50, 35, 12
        DEFPUSHBUTTON "&ตกลง", IDOK, 55, 75, 33, 18,
                WS_CHILD | WS_VISIBLE | WS_TABSTOP
        PUSHBUTTON "&ยกเลิก", IDCANCEL, 117, 75, 33, 18,
        WS_CHILD | WS_VISIBLE | WS_TABSTOP
        CONTROL"ความหนาแน่นของวัสดุ ", -1,"STATIC",SS_RIGHT |SS_NOPREFIX
        | WS_CHILD | WS_VISIBLE | WS_GROUP, 20, 17, 66, 8
        RTEXT "ค่า sphericity", -1, 28, 33, 56, 8, SS_RIGHT |     WS_CHILD | WS_VISIBLE |
                WS_GROUP
        RTEXT "ค่า Volume Shape Factor", -1, 8, 51, 78, 8, SS_RIGHT | WS_CHILD |
                WS_VISIBLE | WS_GROUP
        CONTROL "", -1, "static", SS_BLACKFRAME | WS_CHILD |
                WS_VISIBLE, 6, 7, 179, 61
END
```

Because dialog boxes are frequently executed or created, it makes sense to preserve the latest-values in the dialog's controls for the next time it appears. The transfer buffer type is used for this purpose. This buffer is usually a data member of the parent window. Source code for control the appearance of the dialog box in figure A-3.1 are shown in coding 5.

Coding 5 Source code example for control dialog box with data transfer buffer type.

```
//Define data buffer type
struct TAppTransferBuf1 {
  char DenBuff[MaxEditLen];
  PTComboBoxData DenUnitBuff;
  char SphereBuff[MaxEditLen];
  char ShapeBuff[MaxEditLen];
 };

//Declare MyDialog1 class deriving from TDialog
class MyDialog1 : public TDialog
{
 public:
          MyDialog1(PTWindowsObject AParent, int ResourceId);
          char Shape[20];
          char Density[20];
          char Sphericity[20];
          virtual BOOL CanClose();
 protected:
      void FillBuffers();
          virtual void DenUnit(RTMessage Msg)=[ID_FIRST+ID_DENUNIT];
};

//Constructor of MyDialog1 class
MyDialog1::MyDialog1(PTWindowsObject AParent, int ResourceId)
               : TDialog(AParent, ResourceId)
{
  new TEdit(this, ID_DENSITY, MaxEditLen);
  new TComboBox(this, ID_DENUNIT, MaxEditLen);
  new TEdit(this, ID_SPHERICITY, MaxEditLen);
  new TEdit(this, ID_SHAPE, MaxEditLen);
 TransferBuffer=(void far*)&(((AppWindow *)Parent)->AppBuffer1);
};
```

Note : The declaration of the controls will be arrange in the same order in the resource, the buffer type members, and the dialog box constructor.

Programmers must define the default for dialog box in the Twindow deriving class constructor and declare data buffer type as a public variable in this class. The example of this part is shown in coding 6.

Coding 6 define default for dialog box.

```
class AppWindow : public TWindow
{
  public:
    TAppTransferBuf1 AppBuffer1;
              ...
}

AppWindow::AppWindow(PTWindowsObject AParent, LPSTR ATitle):
          TWindow(AParent, ATitle)
{
              ...

  memset(&AppBuffer1, 0x0, sizeof(AppBuffer1));
  AppBuffer1.DenUnitBuff=new TComboBoxData();
  AppBuffer1.DenUnitBuff->AddString("g/cc     ");
  AppBuffer1.DenUnitBuff->AddString("g/l      ");
  AppBuffer1.DenUnitBuff->AddString("g/cu.m.  ");
  AppBuffer1.DenUnitBuff->AddString("kg/cu.m. ",TRUE);
  AppBuffer1.DenUnitBuff->AddString("kg/l     ");
  AppBuffer1.DenUnitBuff->AddString("lb/cu.ft.");

              ...
}
```

From coding 6, it will appear "kg/cu.m." as a default value in the example combo box and no data (blank) in edit box at the first called.

Combo box data has an index that starts from zero. The command that used in receiving the index value from combo box is shown in coding 7. The receiving index is stored in "denID" variable.

Coding 7 Source code for getting data from combo box.

```
//Choose Density Unit
void MyDialog1::DenUnit(RTMessage Msg)
 {
  DWORD i;
  char str[80];
  ostrstream ostr(str, sizeof(str));
                 if (Msg.LP.Hi==CBN_KILLFOCUS){
                 i=SendDlgItemMsg(ID_DENUNIT,CB_GETCURSEL,0,0L);
                 ostr << i << ends;
                 denID=atoi(str);
                 };
 };
```

For the edit box control, input data are stored as a character and it can be changed to the number by using the function "atof" before closing the dialog box. The function for getting data from edit box is shown in coding 8 as an example. This example is used to get three double variables: density, sphere, and shape. Before closing a dialog, it has a statement that compares these values to zero. This dialog box will not be closed if one of the three is less than zero and a message box will be displayed.

Coding 8 Source code for getting data from edit box.

```
void MyDialog1::FillBuffers()
{
  GetDlgItemText(HWindow, ID_DENSITY, Density, 20);
  GetDlgItemText(HWindow, ID_SPHERICITY, Sphericity, 20);
  GetDlgItemText(HWindow, ID_SHAPE, Shape, 20);
}

BOOL MyDialog1::CanClose()
{
 FillBuffers();

 density=atof(Density);
 sphere=atof(Sphericity);
 shape=atof(Shape);
 if ( density<=0 || sphere<=0 || shape<=0) {
    MessageBox(HWindow, "Data is not Correct","Error",MB_OK);
    return FALSE;
 }
 return TRUE;
}
```

A-3.5 Print Text on the Screen

Programmers can check the input data whether it is correct or not by print them out on screen. The example command is shown in coding 9. It uses the function "sprintf" to get all required data stored in "Str" as string variable. The "TextOut" command is used to print the "Str" variable on the screen at the specify position.

Coding 9 Source code example for printing text on screen.

```
void AppWindow::Paint(HDC DC,PAINTSTRUCT& P)
{ char Str[100];
  int x=15;
  LineHeight=20;
  sprintf(Str,"# ข้อมูลเกี่ยวกับวัสดุที่ใช้  : " );
  TextOut(DC,x,2*LineHeight,Str,strlen(Str));
  sprintf(Str,"ความหนาแน่นของวัสดุ   = %.4f",density);
  TextOut(DC,x+15,3*LineHeight,Str,strlen(Str));
  sprintf(Str,"%s",denunit[denID]);
  TextOut(DC,x+260,3*LineHeight,Str,strlen(Str));
  sprintf(Str,"sphericity                    = %.4f",sphere);
  TextOut(DC,x+15,4*LineHeight,Str,strlen(Str));
  sprintf(Str,"shape factor               = %.4f",shape);
  TextOut(DC,x+15,5*LineHeight,Str,strlen(Str));
}
```

A-4. Creating Definition File

Before compiling the project file, programmers should create the definition file (.DEF). The example for this file is shown in coding 10.

Coding 10 Example of definition file.

```
NAME                    Example
DESCRIPTION             'Example'
EXETYPE                 WINDOWS
CODE PRELOAD MOVEABLE DISCARDABLE
DATA PRELOAD MOVEABLE MULTIPLE
HEAPSIZE                4096
STACKSIZE               5120
```

When all code is finished, programmers must compile this project with all necessary files (.CPP, .RC, and .DEF), if no any error programmers will receive the Windows application desired.

APPENDIX B

HELP FILE DEVELOPMENT

In this research, hypertext technique was used in developed theory section. Hypertext is a term for information that has been divided into separate modules and connected by electronic pathway. Windows Help is one form of hypertext. In developing Help files, the Help topics are created in independently usable modules and they provide numerous forms of access to those topics.

B-1. Tools for Development

In order to develop Windows Help files, several tools are required. These tools are listed as follows.

1. Windows Help (WINHELP.EXE) : It is used for run compiled Help file. This file is included with every copy of Windows.
2. Help Compiler (HC.EXE, HC30.EXE, HC31.EXE) : This compiler is used for converting the topic file (.RTF) into a binary file that is readable by Windows Help.
3. Hot Spot Editor (SHED.EXE) : It allows programmers to create multiple hot spots on a picture. The hot spot can include a jump or a pop-up, or run a Help macro.
4. Word Processor (Word for Windows, AmiPro, WordPerfect, and so on) : A Word Processor is used to create the rich text format files (.RTF) used by the Help compiler. But Word for windows is preferable, because it includes many features that simplify the process of authoring Help.
5. A various packages for develop graphic image (PaintBrush, PhotoShop, and so on) : It is used for creating pictures that required by the Help file.

B-2. The Development Cycle in Creating a Help System

The creation of a Help system for a Windows application comprises the following major tasks:

1. Collecting all information required by the Help system.
2. Planning the Help system.
3. Writing the text and create pictures if required.
4. Entering all required control codes into the text files.
5. Creating a project file for the build.
6. Building the Help file.
7. Testing and debugging the Help system.
8. Programming the application so that it can access Windows Help.

B-3. Coding the Help Topic Files

Before organizing topic files, programmers would choose a Word Processor that can support Rich Text Format (.RTF) and create the footnotes, underlined text, and strike-through or double-underlined text that indicate the control codes.

A topic file may contain one or more topics. Each topic is separated by hard page breaks and can have a title, a list of keywords associated with, a build-tag indicator. Each topic accessible via a hypertext link must have a unique identifier or context string and any topic can have an assigned browse sequence. The topics can also contain optional control codes. Each of the control codes is briefly described in table B-3.1.

Table B-3.1 Topic file control codes.

Control Code	**Purpose**
Hard page break	**Topic separator** – Separate one topic from another.
Pound sign (#) footnote	**Context string** – Define a context string that uniquely identifies a topic.
Dollar sign ($) footnote	**Title** – Determine the title for the topic, as displayed in the Search dialog box, the History list and on the Bookmark menu. Titles are optional.
Plus sign (+) footnote	**Browse sequence number** – Determine the order of a topic in which the user can browse through. It is optional.
Asterisk (*) footnote	**Build tag** – Determine which topics are included when the compiler builds the Help file. Build tags are optional.
Exclamation point (!) footnote	**Macro reference** – Run the specified Help macro when the user displays the topic. Macro references are optional.
Hidden text format	**Cross-reference context string** – Determine context string for the topic that will appear when the user chooses the text that immediately precedes it.
Double-underlined text format	**Jump text** – Indicate the text a user can choose to jump to another topic.
Underlined text format	**Pop-up text** – Specified the text a user can choose to display a pop-up topic.

Note : The method of creating control code as a footnote or formatted text depends on the Word Processor used for creating topic file. For example, Microsoft Word for Windows creates footnote text by choosing “Footnote...” from “Insert” menu then enter the required symbol in the Custom Footnote mark box.

B-4. Entering Control Code into the Text File

In this section, the specific information on using some control codes that used in this research is explained.

B-4.1 Creating a Topic

Each topic in the topic file must be separated with a hard page break. In each topic programmers can insert many control codes and text. The procedure for creating a hard page break depends on the Word Processor used.

B-4.2 Creating a Context String

A context string used to identify a topic must be unique. Context strings enable programmers to create jumps between topics and to display pop-ups. In general, every topic in the topic file should have a context string for accessing through hypertext jumps. If topics do not have context strings, they will be accessible to the user only through browse sequences or a keyword search.

To assign a context string :

1. Place the cursor to the left of the topic heading, then insert a footnote, In Microsoft Word, choose "Footnote..." from the "Insert" menu.
2. Insert the pound sign (#) as a custom footnote. In Microsoft Word, this footnote mark appears as a superscript ($^{\#}$) next to the heading and the footnote pane opens.
3. Type the context string in the footnote pane.

Valid context strings may contain letters (A-Z or a-z), number (0-9), periods (.), and underscore character (_). Context strings are not case-sensitive and in practical its limit is about 255 characters.

B-4.3 Creating a Topic Title

Topic titles appear in Search dialog box, History list and the Bookmark menu. Although not required, most topics have a title. If the topic does not have a title, you will see " >> Untitled Topic << " in the Topics found list.

To assign a title to a topic:

1. Place the cursor to the left of the topic heading, then insert a footnote.
2. Insert a dollar sign ($) as a custom footnote.
3. Type the topic title in the footnote pane. In general, the topic title should match its heading.

Title can be up to 128 characters in length.

B-4.4 Creating a Keyword

The user can search for topics through keywords by calling the Search dialog box. Programmers can not specify a keyword if that topic does not have a title.

To create a keyword :

1. Place the cursor to the left of the topic heading, then insert a footnote.
2. Insert a "K" letter as a custom footnote.
3. Type the keyword, or keywords in the footnote pane. If you add more than one keyword, separate each with a semi-colon (;).

Keywords can include any ANSI character, including accented characters. The maximum length is 255 characters and not case-sensitive.

B-4.5 Creating a Browse Sequence

Browse sequences are accessed using browse buttons that labeled ">>" for moves forward and "<<" for moves backward. These buttons will be disable if that topics are not defined the browse sequences.

To create a browse sequence :

1. Place the cursor to the left of the topic heading, then insert a footnote.
2. Insert a plus sign (+) as a custom footnote.
3. Type the sequence number using alphanumeric characters in footnote pane.

Browse sequences are sorted string in ascending order, for example 0-9 or A-Z.

B-4.6 Creating a Cross-reference or Jump

Cross-reference or jump provides the user with several paths to information. In the color system, Windows Help displays jump, called "hot spots", is identified by green underlined text.

To create a jump :

1. Place the cursor at the point in the topic file where you want to enter the jump term.
2. Change the character formatting to either double-underlined or strike-through format.
3. Type the jump word (or words) in that mode.
4. Turn of the format that used in 3 and select the hidden text attribute.
5. Enter the context string assigned to the topic that is the target of the jump.
6. Change the character formatting back to non-hidden.

B-4.7 Creating a Pop-up

Like a jump, a pop-ups displays information when the user clicks the pop-up with the mouse, but it appears only a small box over the current topic not switching to another topic. Pop-ups display by green, dotted underline text.

To create a pop-up :

1. Place the insertion point in the topic file where you want to place the pop-up term.
2. Change the character formatting to single-underlined format.
3. Type the pop-up term in underlined format.
4. Turn of the underline mode and select the hidden text attribute.
5. Enter the context string assigned to the topic that contains the pop-up term.
6. Change the character formatting back to nonhidden.

B-5. Inserting Graphic Image

Graphic image can be placed in Help topics using two method. First, programmers must paste bitmaps into the topic file if the used Word Processor supports the placement of graphic directly into a document. Second, programmers may place bitmaps by reference. File formats that the Help Compiler support are . BMP (standard Windows bitmap), .DIB (Windows device independent bitmap), . WMF (Windows metafile), .SHG (bitmap containing hotspots created using the Hotspot Editor) and .MRB (multiple-resolution bitmap).

B-5.1 Placing a Bitmap Directly into a File

This method is easiest way to precisely place bitmaps into the topic files. Microsoft Word for Windows support this method by importing directly pictures from the clipboard.

B-5.2 Placing a Bitmap by Reference

To place a bitmap in specific location in the Help topic file, insert one the following statements :

{bmc filename} or {bmcwd filename}

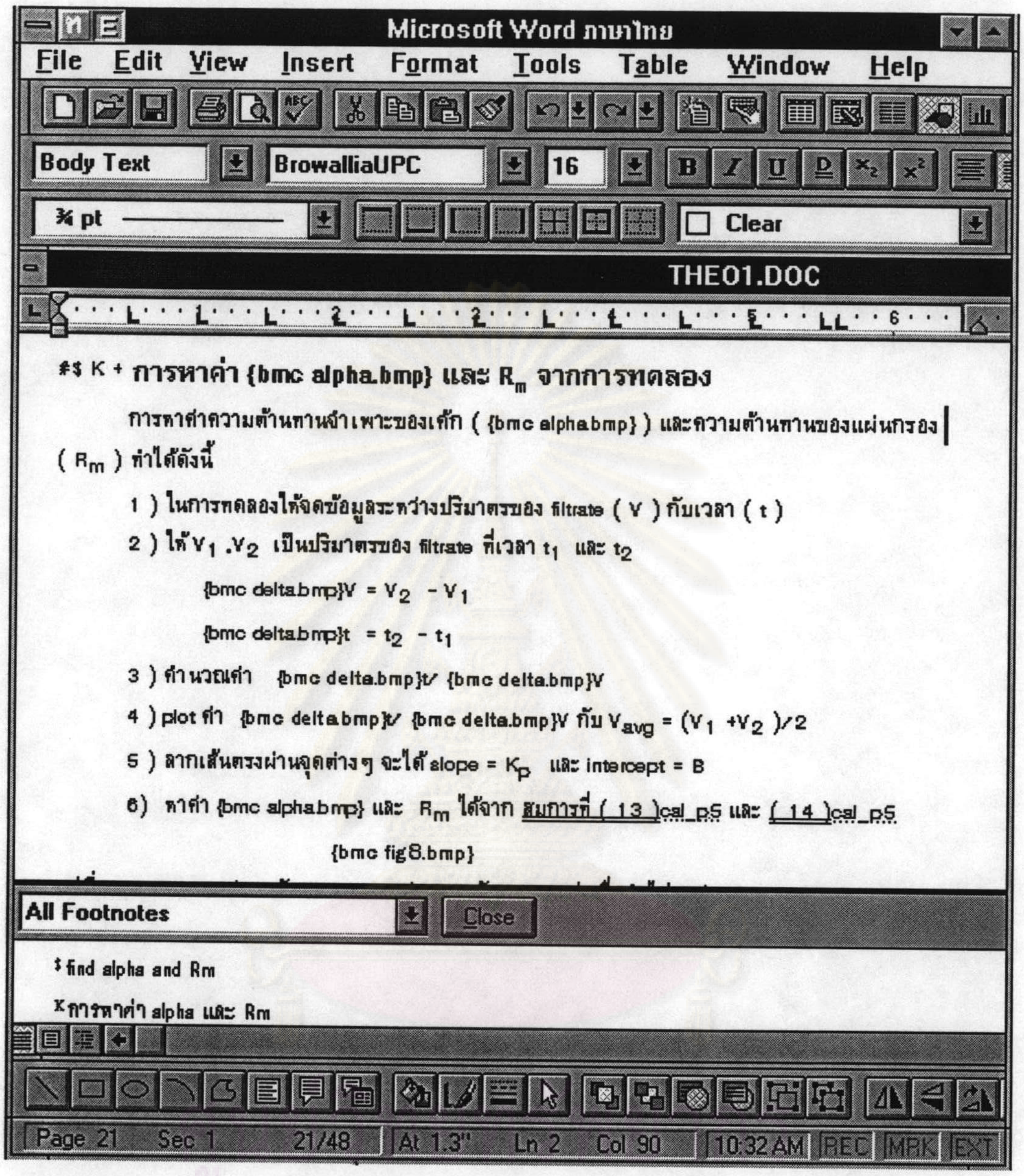
{bml filename} or {bmlwd filename}

{bmr filename} or {bmrwd filename}

where filename is the name of the bitmap file. Each argument indicates the different position of the placing picture. The argument “bmc” or “bmcwd” indicates that the bitmap referred to will be treated the same as a character, so that bitmap will place at the same position of this argument. The argument “bml” or “bmlwd” specify that the bitmap appear at the left margin with text wrapping along its right edge. The argument “bmr” or “bmrwd” is used in showing a bitmap on the right margin, with text wrapping along its left edge.

Programmers can create a multiple hotspots in a bitmap by using Hotspot Editor and save as a Hypergraphic file (.SHG).

The example for Help topic file with inserting bitmap by reference is shown in figure B-5.1.


Microsoft Word ภาษาไทย
File Edit View Insert Format Tools Table Window Help
Body Text
BrowalliaUPC
16
¾ pt
Clear
THE01.DOC
All Footnotes
Close
Page 21 Sec 1 21/48 At 1.3" Ln 2 Col 90 10:32 AM REC MRK EXT

#$ K + การหาค่า {bmc alpha.bmp} และ R_m จากการทดลอง

การหาค่าความต้านทานจำเพาะของเค้ก ({bmc alpha.bmp}) และความต้านทานของแผ่นกรอง (R_m) ทำได้ดังนี้

1) ในการทดลองได้จดข้อมูลระหว่างปริมาตรของ filtrate (V) กับเวลา (t)

2) ให้ V_1 ,V_2 เป็นปริมาตรของ filtrate ที่เวลา t_1 และ t_2

{bmc delta.bmp}V = $V_2 - V_1$

{bmc delta.bmp}t = $t_2 - t_1$

3) คำนวณค่า {bmc delta.bmp}t/ {bmc delta.bmp}V

4) plot ค่า {bmc delta.bmp}t/ {bmc delta.bmp}V กับ $V_{avg} = (V_1 + V_2)/2$

5) ลากเส้นตรงผ่านจุดต่างๆ จะได้ slope = K_p และ intercept = B

6) หาค่า {bmc alpha.bmp} และ R_m ได้จาก สมการที่ (13)cal_p5 และ (14)cal_p5

{bmc fig8.bmp}

$ find alpha and Rm

K การหาค่า alpha และ Rm

Figure B-5.1 Example for Help Topic file with inserting a bitmap by reference.

B-6. Creating the Project File (*.HPJ)

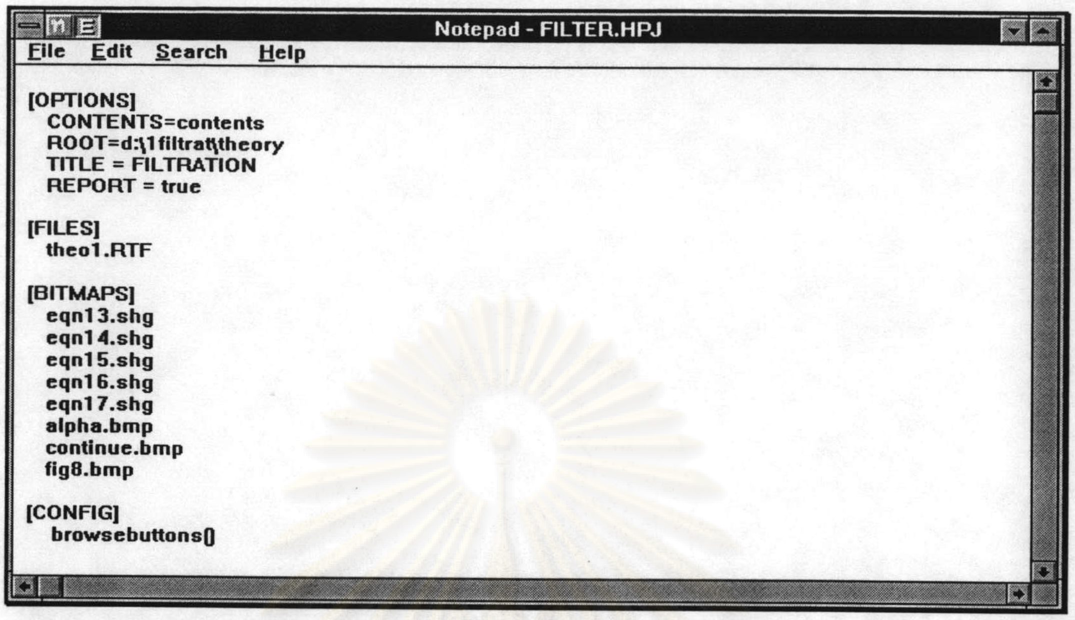
Project file can be created by using text editor or Word Processor, then it can be saved as unformatted text with a .HPJ file extension. This file functions as an interface between Help topic file and the Help Compiler.

The Help project file can contain up to nine sections that perform the following functions in table B-6.1.

Table B-6.1 Help project file sections

Section	Function
[OPTIONS]	Specifies various options that contain how a Help file is built. Such as the level of error report, topics to be included in the build and so on. This section is optional.
[FILES]	Specifies the topic files included in the build. This section is required.
[BUILDTAGS]	Specifies valid buildtags. It is optional.
[CONFIGS]	Specifies any customized menu, button and the registers Dynamic Link Library (.DLL) in Help file. This section is optional.
[BITMAPS]	Specified bitmap files included in the build. This section is optional.
[MAP]	Associate context strings with context numbers. This section is optional.
[ALIAS]	Assign one or more context strings to the same topic. It is optional.
[WINDOWS]	Controls the appearance of the main window and secondary windows. It is optional.
[BAGGAGE]	Lists files that you want to place within Help's .HLP file. It is optional.

A sample of simple project file is illustrated in figure B-6.1. There are only four sections in this project file.



```
[OPTIONS]
  CONTENTS=contents
  ROOT=d:\1filtrat\theory
  TITLE = FILTRATION
  REPORT = true

[FILES]
  theo1.RTF

[BITMAPS]
  eqn13.shg
  eqn14.shg
  eqn15.shg
  eqn16.shg
  eqn17.shg
  alpha.bmp
  continue.bmp
  fig8.bmp

[CONFIG]
  browsebuttons()
```

Figure B-6.1 A sample project file (.HPJ).

B-7. Compiling a Help File

After creating Help Project file, programmers are ready to compile .HPJ file. To run the Help Compiler, one can use the "HC" command.

The "HC" command uses the following syntax :

```
HC path\filename.HPJ
```

The result of compiling is a Help resource file with .HLP extension in the current directory. The name of the result file is the same as that of Help Project file.

B-8. Programming the Application to Access the Help File

This work used of Turbo C++ for Windows as a compiler to access this application. It can access the Help resource file by calling the "WinHelp" function that uses the following syntax:

```
BOOL WinHelp(hWnd, lpHelpFile, wCommand, dwData);
```

Each parameter specifies these:-

- The hWnd parameter identifies the Windows requesting Help.

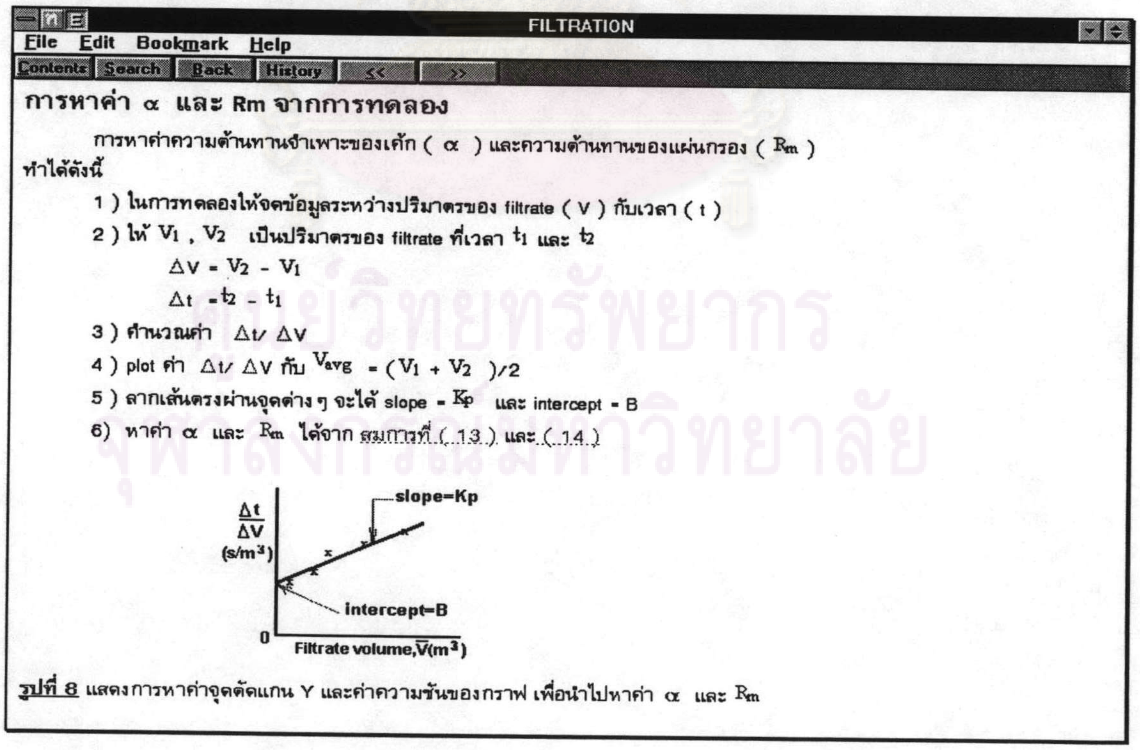
- The lpHelpFile parameter specifies the name (with optional directory path) of the Help file containing the desired topic.
- The wCommand parameter specifies the type of search the Windows Help application is used to locate the specified topic.
- The dwData parameter specifies the topic for which the application is requesting Help.

For example, filter.hlp file will show after it receives CMTheory message:

```
void AppWindow::CMTheory(RTMessage)
{
      WinHelp(HWindow,"filter.hlp",HELP_INDEX,0L);
}
```

B-9. Help File Example

After compiling the Help project file, the Help file with hotspots and hypertext link will be built as shown in figure B-9.1.



การหาค่า α และ Rm จากการทดลอง

การหาค่าความต้านทานจำเพาะของเค้ก (α) และความต้านทานของแผ่นกรอง (R_m) ทำได้ดังนี้

1) ในการทดลองให้จดข้อมูลระหว่างปริมาตรของ filtrate (V) กับเวลา (t)

2) ให้ V_1 , V_2 เป็นปริมาตรของ filtrate ที่เวลา t_1 และ t_2

$\Delta V = V_2 - V_1$

$\Delta t = t_2 - t_1$

3) คำนวณค่า $\Delta t / \Delta V$

4) plot ค่า $\Delta t / \Delta V$ กับ $V_{avg} = (V_1 + V_2)/2$

5) ลากเส้นตรงผ่านจุดต่าง ๆ จะได้ slope = K_p และ intercept = B

6) หาค่า α และ R_m ได้จาก สมการที่ (13) และ (14)

$\frac{\Delta t}{\Delta V}$ (s/m³)
slope=Kp
intercept=B
0
Filtrate volume,$\overline{V}$(m³)

รูปที่ 8 แสดงการหาค่าจุดตัดแกน Y และค่าความชันของกราฟ เพื่อนำไปหาค่า α และ R_m

Figure B-9.1 An example of Help file .

APPENDIX C

LINEAR REGRESSION METHOD

The collecting data from laboratory may be inconsistency because of errors in measurement. "Least squares estimation" is one way of an approximate solution. This method fits the data point by selecting the parameters so that the sum of squares of the errors between the data points and the curve is minimal. Here, in the work, only linear relationships involving two variables is examined.

The equation for a straight line where the dependent variable Y is determined by the independent variance X is

$$Y = a + bX \qquad \text{(C-1)}$$

The value of Y can be computed from a given value of X by using this equation. The a is called the "Y-intercept" because its value is the point at which the regression line crosses the Y-axis. The b is a "slope" of the line. Both a and b are numerical constants. For any given straight line their values do not change.

A new symbol Y' is introduced to represent the individual values of the estimated points. Accordingly, we shall write the equation for the estimating line as

$$Y' = a + bX \qquad \text{(C-2)}$$

For this method, we assume that n pairs of X and Y values have been observed. We denote these pairs as (X_1,Y_1), (X_2,Y_2), ..., (X_n,Y_n).

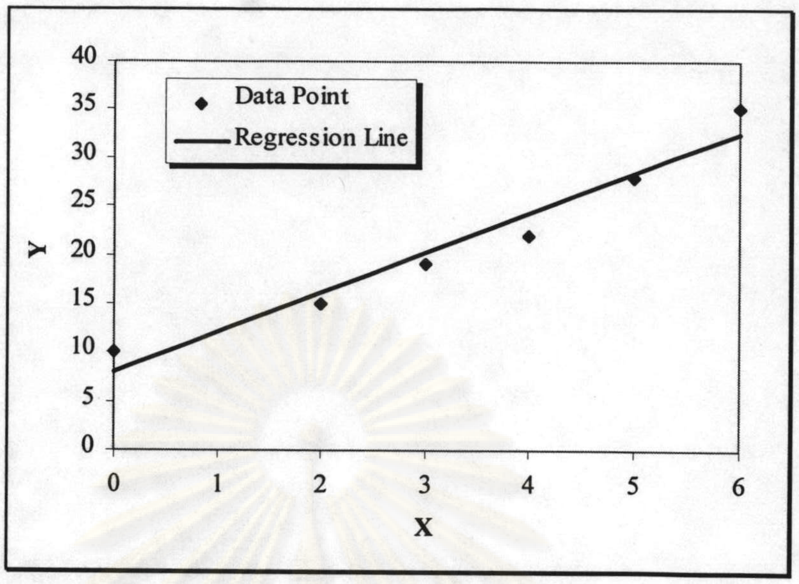


Figure C-1 Regression forecasting.

Error between the observe value and the predicted values is defined as

$$e_i = Y_i' - Y_i \qquad \text{(C-3)}$$

And the total variance or squared error due to all the points is

$$\sum e_i^2 = \sum (Y_i' - Y_i)^2 = \sum (a + bX_i - Y_i)^2 \qquad \text{(C-4)}$$

In regression analysis, the objective is to minimize the error equation shown above by choosing value of a and b. The minimum error can be found by solving the following set of equations simultaneously:

$$a = \frac{\sum Y_i}{n} - b\frac{\sum X_i}{n} \qquad \text{(C-5)}$$

$$b = \frac{n\sum X_i Y_i - (\sum X_i)(\sum Y_i)}{n\sum X_i^2 - (\sum X_i)^2} \qquad \text{(C-6)}$$

With these two equations, we can find the best-fitting regression line for any two-variable set of data points.

We can also calculate the confidence of the relationship between Y and X by means of r, the correlation coefficient. However, r^2, the coefficient of determination, has a direct physical meaning, which r does not have, so it is better to use r^2. The value of r^2 represents the proportion of variation in Y which is explained by the relationship with X. On the other hand, the variation $(1-r^2)$ is due to chance or factors other than X. For example, if a value of $r^2 = 0.8$, it will indicate that 80 percent of the variation in Y is predicted by the regression line with X; only 20 percent is due to chance.

The quantity r^2 can be computed as follows:

$$r^2 = \frac{\left[n\sum X_iY_i - \left(\sum X_i\right)\left(\sum Y_i\right)\right]^2}{\left[n\sum X_i^2 - \left(\sum X_i\right)^2\right]\left[n\sum Y_i^2 - \left(\sum Y_i\right)^2\right]} \qquad \text{(C-7)}$$

The r^2 value can be used to measure the reliability of the estimating equation. The smaller the value of r^2, the wider points scatter around the regression line. Conversely, if $r^2=1$, we expect the estimating equation to be a "perfect" representative of the dependent variable. In that case, all the data points would lie directly on the regression line, and no point scatters around it. On the other hand, there is no correlation when the value of $r^2=0$.

VITA

Miss Napaporn Ratanapoka was born in Patumthani, Thailand on September 16, 1972. She received her Bachelor Degree of Science with first class honors from Department of Chemistry, Faculty of Science, Kasetsart University in 1993.