Quick Start

MATFOR
In Visual C#

ANCAD INCORPORATED

TEL: +886(2) 8923-5411
FAX: +886(2) 2928-9364
support@ancad.com
www.ancad.com

Quick Start: MATFOR Version 4.1

Contents

Chapter 1

INTRODUCTION

This Quick Start gives a brief overview of using the different components in MATFOR version 4.0. Language considerations, the structure of programs, and the use of advanced tools are all covered.

I. PRODUCT DESCRIPTION

MATFOR is a set of numerical and graphical libraries developed to enhance computation and visualization in different programming environments: C++, Fortran, Visual Basic and Visual C#. Especially designed for scientists and engineers, MATFOR aims to accurately perform computation, dynamically visualize data, and efficiently decrease process time. Its features include:

mfArray integrates the entire MATFOR toolkit into high-level programming environments such as C++, Fortran, Visual Basic and Visual C#, simplifying the syntax and facilitating object-oriented programming.

Numerical Library contains useful linear algebraic functions subject to assist users with computational problem solving.

Visualization Library collects well-designed graphical procedures and controls to support a variety of 2D and 3D visual functions.

Data Viewer, organized in spreadsheet format, is one convenient platform for data management, filter, and analysis.

Graphics Viewer, besides its highly customized user interface, overthrows the convention of post-processing as it instantly visualizes scientific and engineering data.

Graphics Export converts dynamic presentation into standalone movie or image files to enhance accessibility of simulation results.

MATFOR Control facilitates application-building by integrating its graphic component, MatforWidget, into VB and C# environments.

II. COMPILER REQUIREMENTS

MATFOR supports these compiler choices:

- Visual C# 2005

III. INSTALLATION

This section provides step-by-step instructions for installing MATFOR. If you encounter any problems during the installation, please contact support@ancad.com.

- **Pre-Installation**
  1. Exit any of MATFOR programs executing;
  2. Remove all previous versions of MATFOR components if this is an upgrade;
  3. Ensure the compiler requirements are satisfied;
  4. Obtain administrator rights under Windows 98/2000/NT/XP.

- **Begin Installation**
  1. Insert the MATFOR CD into the CD-ROM drive.

     The standard MATFOR Installation Procedure shall start automatically. If the Procedure fails to start, you may manually start it by double-clicking on the MATFOR.exe file under the **<CDROM>\Content**\ path.
  2. Follow the self-explanatory instructions in the Procedure to set up all MATFOR components.
  3. Specify a destination folder to create MATFOR through the Procedure.

     By default, the Package creates a program group in your C drive under the path **C:\Program Files\AnCAD\MATFOR4**.

- **Set Environment Path**

  The Procedure automatically sets the $PATH environment variable.

  To manually set the environment variables:
  1. Go to **Control Panel\System**.
  2. In the **System Dialog Box**, select the Advance label and click on the Environment

Variables tab.

3. In the **System Variables Box**, add **<MATFOR4>\bin** and **<MATFOR4>\tools** to the [path] variable.

   NOTE

   <MATFOR4> shall specify the directory under which MATFOR is installed.

   i.e. C:\Program Files\AnCAD\MATFOR4

IV. REGISTRATION

At the end of the installation, you are required to enter the License Password for registration, which can only be obtained by submitting the **Host ID** of your local machine to us: (For trial users, please obtain the License Password by sending email to sales@ancad.com.)

1. Go to http://www.ancad.com/activation.php to activate MATFOR with the **Serial Number** and **Host ID** information.
2. MATFOR license key should be sent to you through the email provided within 24 hours.

V. FLOATING LICENSE & CLASSROOM LICENSE

To enable a group of developers to share a pool of licenses more efficiently, MATFOR offers floating licenses and classroom licenses for cross-system/cross-platform environments. The floating license is designed to be used in any shared network environment; the classroom license is designed for the purpose of teaching in an academic environment. Any project developed under MATFOR classroom license may not be redistributed to a third party with profit interest or being a commercial institution.

These license models consist of one or more license servers; the license server runs the license management process and monitors number of concurrent users using MATFOR licenses in the network. The installation steps are:

- **Installation of License Servers**

  1. Run **license_server.exe** (download from http://www.ancad.com/activation.php) on machines designated as license servers.
  2. Retrieve required information from the machine.
     Under command prompt,

i. Change directory to ***C:\Program Files\AnCAD\License Server***.

ii. Execute ***lmtools*** to get Ethernet Address and Host Name.

3. Collect the following product information:
   i. Compiler version
   ii. Operating system
   iii. Number of license of each version
4. Obtain license file(s) for license server(s).
   i. Go to http://www.ancad.com/activation.php to activate MATFOR with the **Serial Number** and **Host ID** information.
   ii. The license file(s) should be sent to you through the email provided within 24 hours.
5. Place the license file **xxx.lic** into ***<AnCAD>\License Server*** directory.
6. Start the license server.

   Under command prompt, type:

   lmgrd –c *xxx.lic* or

   lmgrd –c . for multiple license files.

- **Configuration in Client Computers**
  1. Install MATFOR on machines designated as clients.
  2. Go to **Start ▶ Program Files ▶ MATFOR4 ▶ Utilities ▶ Register MATFOR.**
  3. Under Registration Window, choose Network License.
  4. Fill in the Host Name or IP Address of the license server in the blank area. (IP Address can be obtained from the license server by typing ***lmhostid -internet –n*** under command prompt.)

For other questions regarding MATFOR floating license or classroom license installation and registration, please contact support@ancad.com.

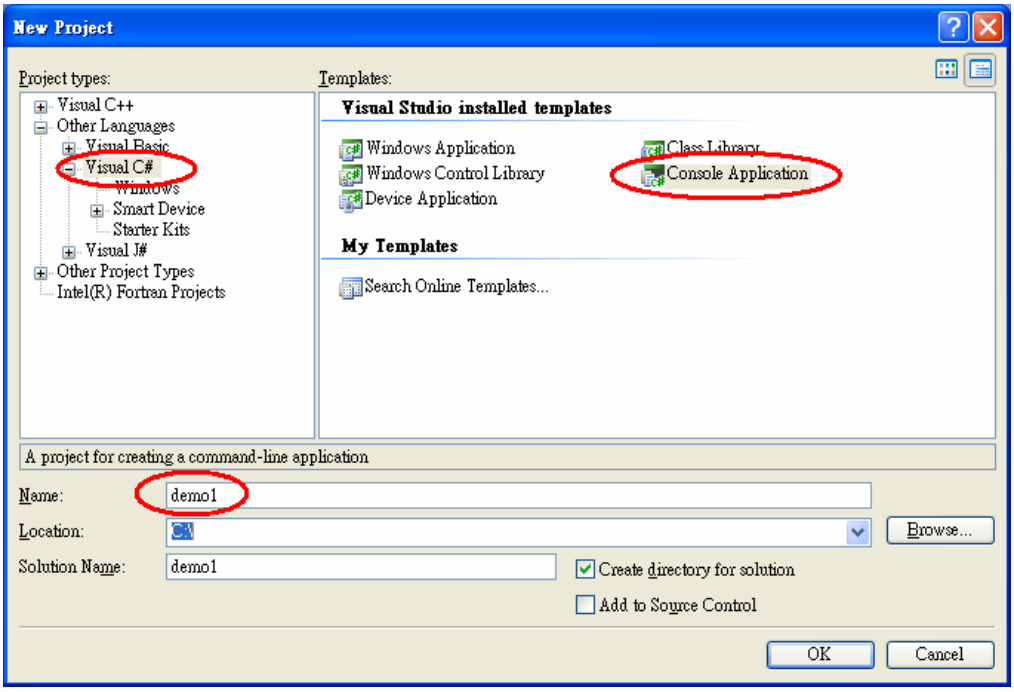
Chapter 2

ENVIRONMENT SETTING

As a library, manual configuration is required to use MATFOR under different compiler environments. This chapter is composed to guide users to configure project setting under Microsoft Visual C#.

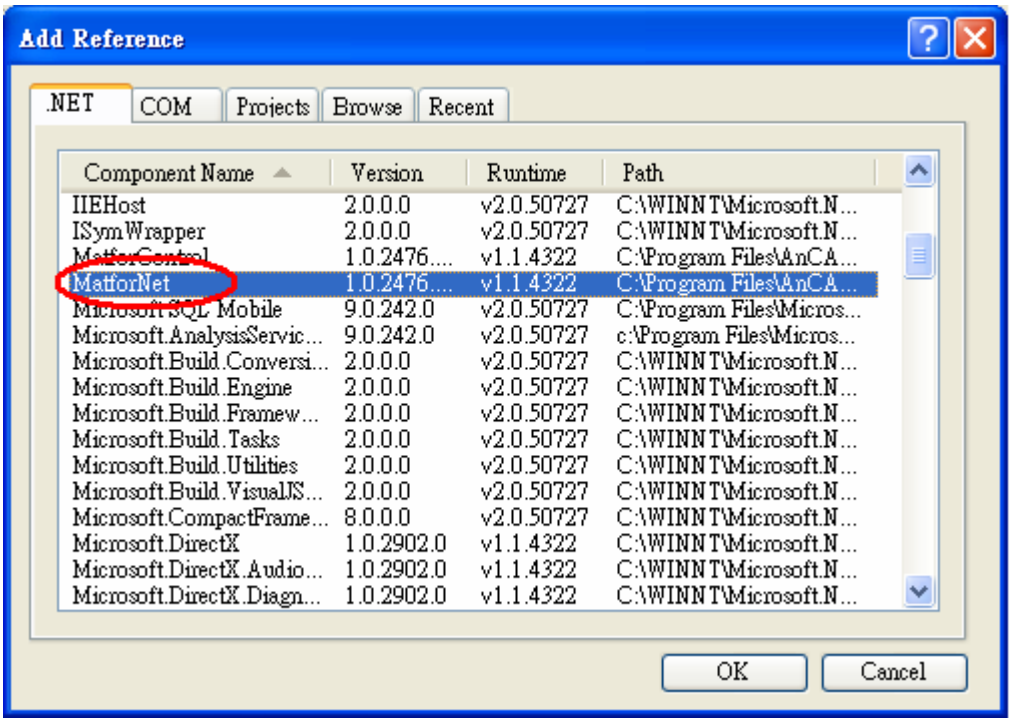
I. MICROSOFT VISUAL C#

1. Open Microsoft Visual Studio 2005.
2. Go to **File** ▶ **New** ▶ **Project**.
3. Select **Visual C#** from the **Project types** menu.
4. Select **Console Application** from **Templates**.
5. Enter a name for the new file, specify its location, and click **OK**.



6. Select **Add Reference** from the **Project** menu.

7. Select **MatforNet** from the Component Name menu under the **.NET** tab and click **OK**.



Chapter 3

FUNDAMENTALS

This chapter is an introductory summary on how to integrate MATFOR into C# programming. Beginning with mfArray, the core of MATFOR, the chapter subsequently introduces the numerical library and the visualization library.

I. MFARRAY

mfArray is a dynamic array endorsing the data types and dimensions listed in Table 3.1.

Data Type	Statement
string	a = “string’”
boolean	a = true
int	a = 1
float	a = 1.0f
double	a = 3.1418f

Table 3.1 | mfArray Data Types

mfArray Declaration

1. Include MATFOR library to appropriately contain mfArray.
2. Declare mfArray using the format ***mfArray <variable>***

Example Code (ch3-1)

```
using System;
using MatforNet;

namespace demo
{
    class demo1
    {
        [STAThread]
        static void Main(string[] args)
        {
                // Declare mfArray variables x and y
                mfArray x, y ;
                // x is a scalar
                x = 6.0 ;
                // y is a 1-by-five vector containing real values
                y = mf.V(1.0, 2.0, 3.0, 4.0, 5.0) ;
                // Call mfDisplay function to display x and y data
                mf.Display(x, "x", y, "y");
        }
    }
}
```

Result

```
x =

 6

y =

 1  2  3  4  5
```

II. NUMERICAL PROCEDURES

MATFOR embraces a comprehensive set of numerical procedures furnished with easy-to-call syntax to complement mfArray. Various categories of numerical procedures are contained in the set and are hereto listed in Table 3.2.

For a complete list of the numerical routines and descriptions on their usage and functionality, please refer to the Reference Guide.

Procedure Group	Example	Description
Data Manipulations	mf.Sort	Sort data in ascending order
Arithmetic Operators	mf.RDiv	Right-divide the matrix
Trigonometry	mf.Cos	Request cosine function
Exponential	mf.Log	Request natural logarithm
Complex	mf.Conj	Generate conjugate of complex
Rounding and Remainder	mf.Ceil	Round towards positive infinity
Matrices	mf.Magic	Construct magic square
Matrix Manipulations	mf.Find	Find indices of nonzero elements
Matrix Analysis	mf.Norm	Generate matrix/vector norm
Linear Equations	mf.Chol	Request Cholesky factorization
Eigenvalues and Singular Values	mf.Schur	Perform Schur decomposition
Factorization Utilities	mf.Balance	Perform diagonal scaling

Table 3.2 | Numerical Procedure Brief

Numerical Procedure Call

Use “MatforNet” to appropriately contain the numerical routine(s).

Example Code (ch3-2)

```
using System;
using MatforNet;

namespace demo
{
    class demo2
    {
        [STAThread]
        static void Main(string[] args)
        {
            // Declare mfArray variables m, i, and j
            mfArray m, i, j ;

            // Construct a 3-by-3 magic matrix and assign it to m
            m = mf.Magic(3) ;

            // Retrieve the row-column indices of the nonzero(s) in m
            // i and j describe the row and column, respectively
            mf.Find(mf.Out(out i, out j), m);


            mf.Display(m) ;              // Display matrix m
            // Call mfDisplay function to display x and y data
            mf.Display(i, "i", j, "j");
        }
    }
}
```

Result

```
ans =

 8  1  6
 3  5  7
 4  9  2

i =

 1
 2
 3
 1
 2
 3
 1
 2
 3

j =

 1
 1
 1
 2
 2
 2
 3
 3
 3
```

III. VISUALIZATION PROCEDURES

In addition to the numerical procedures, MATFOR endorses a collection of visualization routines to enhance data comprehension. Table 3.3 summarizes the procedures and introduces an example routine with description for each procedure group.

Procedure Group	Example	Description
Window Management	mf.WindowSize	Set the frame size of Graphics Viewer window
Visualization Controls	mf.Subplot	Create subplot in active figure
Graph Options	mf.Surf	Construct 3D surface plot
Advanced Graph Options	mf.GetDelaunay3	Construct 3D Delaunay triangulation
Object Manipulations	mf.ObjOrigin	Set the origin of the drawn object
Appearance Settings	mf.Colormap	Specify the colormap type of the drawn object
Annotations	mf.Title	Annotate the graph with title
3D Objects	mf.Sphere	Draw a sphere

Table 3.3 | Visualization Procedure Brief

For a complete list of the visualization routines and descriptions on their usage and functionality, please refer to the Reference Guide.

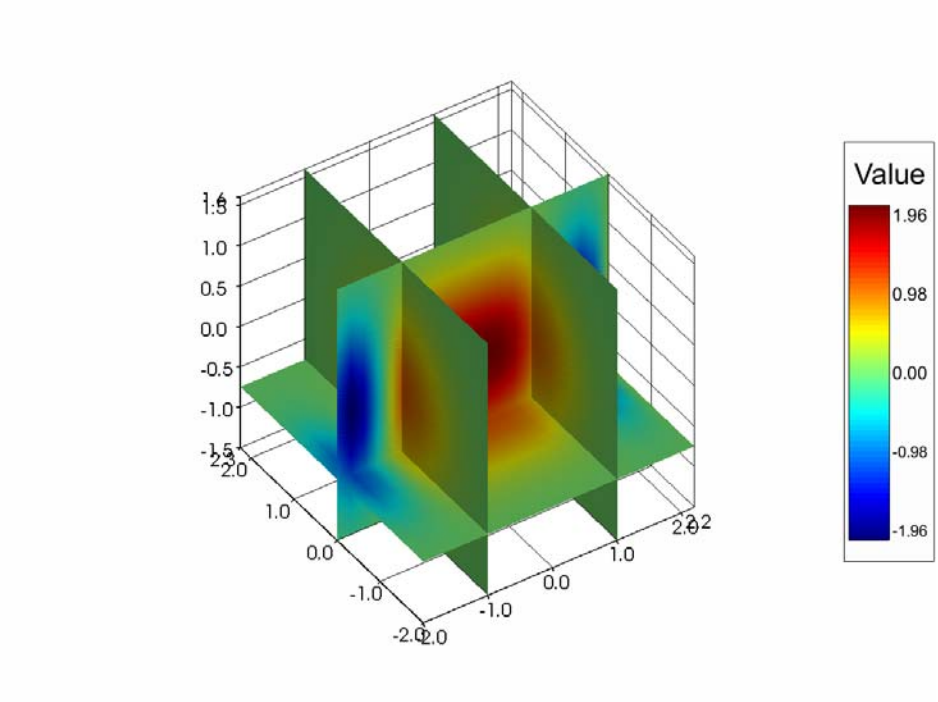
Visualization Procedure Call

Use “MatforNet” to appropriately contain the visualization routine(s).

Example Code (ch3-3)

```
using System;
using MatforNet;

namespace demo
{
    class demo3
    {
        [STAThread]
        static void Main(string[] args)
        {
            mfArray nx, ny, nz;

            // Declare mfArray variables
            mfArray x, y, z, c, tet;

            // Construct linearly spaced vector nx
            nx = mf.Linspace(-2, 2.2, 21);
            // Construct linearly spaced vector ny
            ny = mf.Linspace(-2, 2.25, 17);
            // Construct linearly spaced vector nz
            nz = mf.Linspace(-1.5, 1.6, 31);
            // Construct grid matrices for 3D plotting
            mf.Meshgrid(mf.Out(out y, out x, out z), ny, nx, nz);

            // Mathematically define c
            c = 2 * mf.Cos(x*x) * mf.Exp( - (y*y) - (z*z));

            // Construct 3D Delaunay triangulation
            tet = mf.GetDelaunay3(x, y, z);

            // Display orthogonal sliced planes
            mf.TetSliceXYZ( tet, x, y, z, c,mf.V(-1.0, 1.0), 0, -0.75 );

            // Pause to display the graph
            mf.ViewPause();
        }
    }
}
```

Result


Value
1.96
0.98
0.00
-0.98
-1.96
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
2.0
1.0
0.0
-1.0
-2.0
-2.0
-1.0
0.0
1.0
2.0

Chapter 4

A FIRST PROGRAM

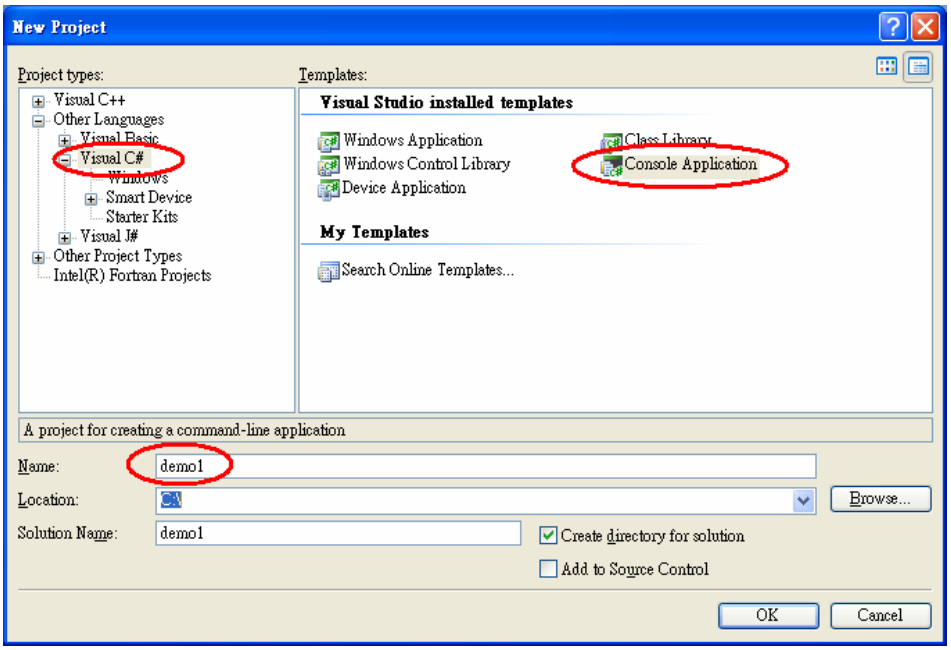
This chapter presents one simple visualization program for users to manipulate and extend as animation and recording procedures are introduced.

I. HELLO SURFACE

The section shall take you step by step to create a surface plot.

Create a New Project

1. Select **Console Application** from **Templates**.
2. Name the project "**demo1**," specify its location, and click **OK**.



Adding the Source Code (ch4-1)

In "**Program.cs**," type in:

```
using System;
using MatforNet;

namespace demo1
{
    class Program
    {
        [STAThread]
        static void Mainstring[] args)
        {
            // Declare mfArray variables
            mfArray x, y, z;

            // Construct grid matrices for 3D plotting
            mf.Meshgrid(mf.Out(out x, out y),
                        mf.Linspace(-3, 3, 30),
                        mf.Linspace(-3, 3, 30));

            // Mathematically define z
            z = mf.Sin(x) * mf.Cos(y);

            // Plot a surf using mfArray x, y, and z
            mf.Surf(x, y, z);

            // Pause to display the graph
            mf.ViewPause();
        }
    }
}
```

Line-by-Line Walkthrough

using MatforNet;

This includes the library file MatforNet which contains the MATFOR numerical and visualization procedures.

mfArray x, y, z;

This declares mfArray variables x, y, and z.

mf.Meshgrid(mf.Out(out x, out y), mf.Linspace(-3, 3, 30), mf.Linspace(-3, 3, 30));

This creates x and y grid matrices from the domains specified by Linspace-generated vectors.

- mf.Out(out x, out y) specifies x and y as mfArray outputs.

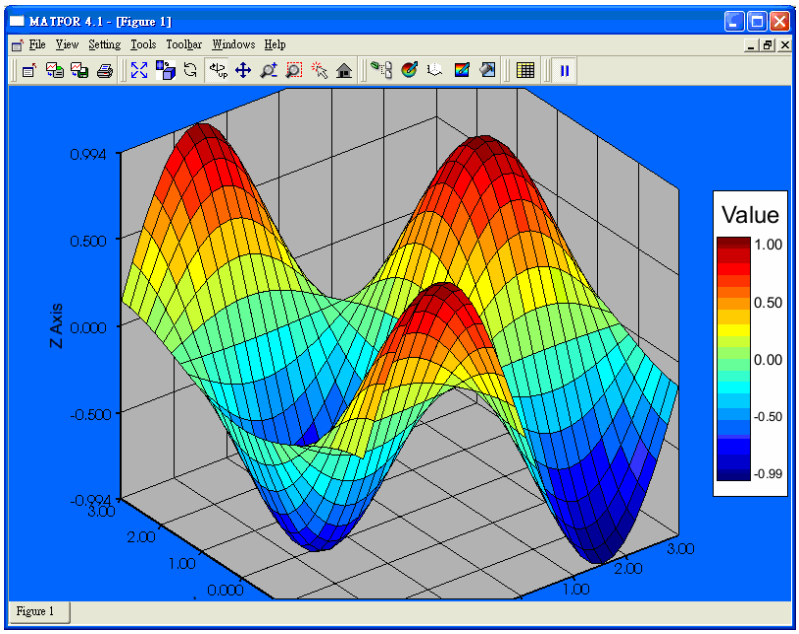
- mf.Linspace(-3, 3, 30) constructs a vector with 30 linearly spaced points between -3 and 3.

z = mf.Sin(x) * mf.Cos(y);
This defines z mathematically.

mf.Surf(x, y, z)
This creates a three-dimensional surface plot of the x-, y-, and z-coordinates.

mf.ViewPause();
This halts program execution for graphical display. mf.ViewPause() should be added after each set of graphical creation routines.

Compile and Run



II. TO ANIMATE HELLO SURFACE

In this section, the Hello Surface demo code is modified to generate animation.

Animation

Animations effects are produced by continuously updating data displayed in the Graphics Viewer.

To Animate Data

1. Use MatforNet to appropriately contain the numerical and visualization routine(s).
2. Construct and initialize the mfArrays for plotting.
3. Create a static plot of the graph to be animated.
4. Set up an iteration loop for the range of data to be observed through animation.
5. Within the loop, use procedure *mf.GSet(handle, 'axis-data', data)* to update the targeted data of the current draw.
6. Update the drawn figure accordingly by using procedure *mf.DrawNow*.
7. Use procedure *call mf.ViewPause* after the end of animation to observe the static graph.

Example Code (ch4-2)

Modify the previously created "**Program.cs**" file as follows and rename it "**Program2.cs**":

```
using System;
using MatforNet;

namespace demo1
{
    class Program2
    {
        [STAThread]
        static void Main(string[] args)
        {
            // Declare mfArray variables
            mfArray x, y, z, h;
            // Declare an integer variable
            int i;
            // Construct grid matrices for 3D plotting
            mf.Meshgrid(mf.Out(out x, out y),
                        mf.Linspace(-3, 3, 30),
                        mf.Linspace(-3, 3, 30));

            h = 0;              // Define h
            for (i = 1; i <= 30; i++)         // Create loop
            {
                // Mathematically define z
                z = mf.Sin(x + i / 8.0) * mf.Cos(y);
                if (i == 1)
                {
                    // Plot a surf using mfArrays x, y, and z
                    h = mf.Surf(x, y, z);
                    mf.DrawNow();
                }
                else
                {
                    // Update the figure as the z-coordinate varies
                    mf.GSet(h, "zdata", z);
```

```
                    mf.DrawNow();
                }
            } // End loop
            // Pause to display the graph
            mf.ViewPause();
        }
    }
}
```

Line-by-Line Walkthrough

using MatforNet;

This includes the library file MatforNet which contains the MATFOR numerical and visualization procedures.

mfArray x, y, z, h;

This declares mfArray variables x, y, z, and h.

int i;

This declares an integer variable i.

mf.Meshgrid(mfOut(out x, out y), mf.Linspace(-3, 3, 30), mf.Linspace(-3, 3, 30))

This generates x and y grid matrices from the domains specified by the mf.Linspace-generated vectors. This procedure aims to solve functions for two variables by plotting 3D graphs.

for (i = 1; i <=30; i++)

This creates a loop running from i=1 to i=30.

z = mf.Sin(x+i/8.0) * mf.Cos(y);

This defines z mathematically. Note that the equation involves the loop index i.

if (i == 1) {

At the beginning of the loop,

h = mf.Surf(x, y, z);

This creates a three-dimensional surface plot of the x-, y-, and z-coordinates.

mf.DrawNow();

This displays the initial figure.

else

At all other times during loop execution,

mf.GSet(h, "zdata", z);

This sets the z-coordinate for update.

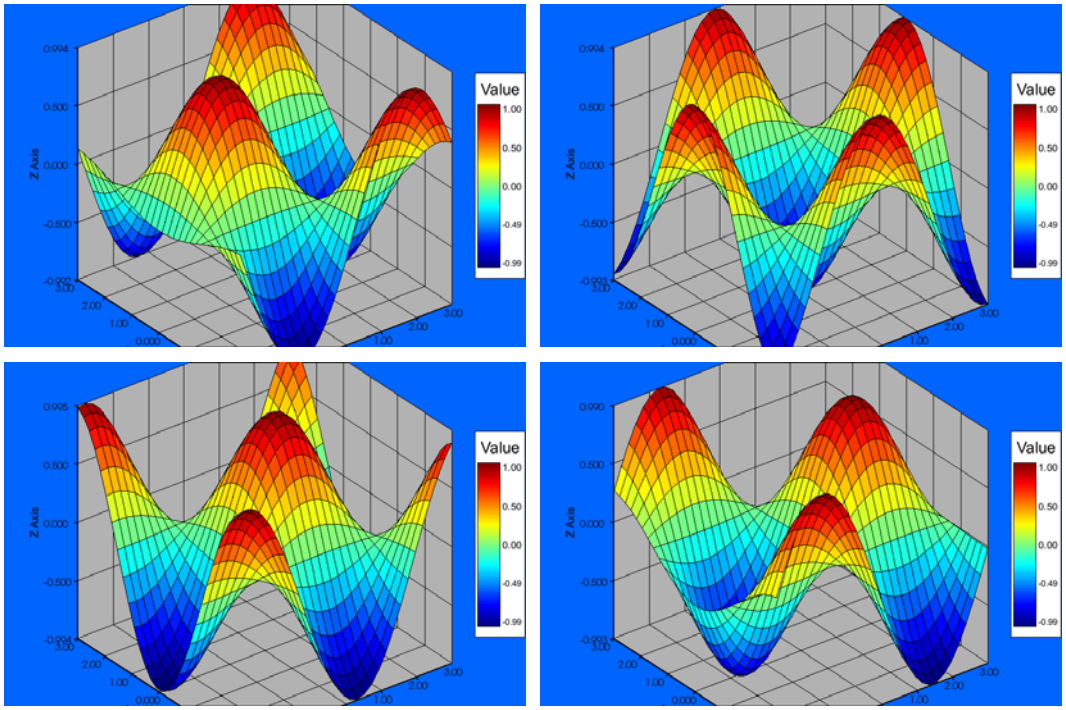
mf.DrawNow();

This displays and updates the figure during loop execution.

mf.ViewPause();

This halts program execution for graphical display.

Compile and Run



III. TO RECORD HELLO SURFACE

This section revises the previous sample code to perform animation recording.

Recording

MATFOR allows recording of animated simulation to facilitate date presentation with comprehensiveness and readability.

To Record Animation

1. Use MatforNet to appropriately contain the numerical and visualization routine(s).

2. Use procedures **mf.RecordStart("animation.avi")** and **mf.RecordEnd()** before and after the animation codes to record the animation.

Example Code (ch4-3)

Add the recording routines to the previously created "**Program2.cs**" file and rename it "**Program3.cs**" (Note: only the modified and added lines are shown in colors):

```
using System;
using MatforNet;

namespace demo1
{
    class Program2
    {
        [STAThread]
        static void Main(string[] args)
        {
            // Declare mfArray variables
            mfArray x, y, z, h;
            // Declare an integer variable
            int i;
            // Construct grid matrices for 3D plotting
            mf.Meshgrid(mf.Out(out x, out y),
                        mf.Linspace(-3, 3, 30),
                        mf.Linspace(-3, 3, 30));

            mf.RecordStart("demo.avi");       // Begin recording

            h = 0;              // Define h
            for (i = 1; i <= 30; i++)         // Create loop
            {
                // Mathematically define z
                z = mf.Sin(x + i / 8.0) * mf.Cos(y);
                if (i == 1)
                {
                    // Plot a surf using mfArrays x, y, and z
                    h = mf.Surf(x, y, z);
                    mf.DrawNow();
                }
                else
                {
                    // Update the figure as the z-coordinate varies
                    mf.GSet(h, "zdata", z);
                    mf.DrawNow();
                }
            } // End loop

            mf.RecordEnd();   // End recording
```

```
                // Pause to display the graph
                mf.ViewPause();
            }
        }
}
```

Chapter 5

THE ADVANCED FEATURE

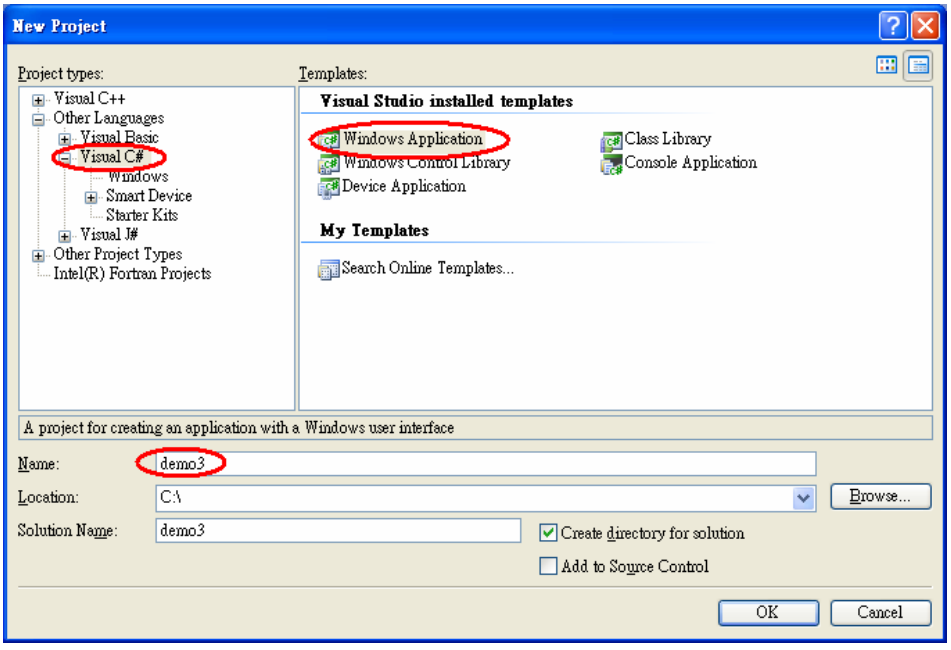
This chapter focuses on the graphic components of MATFOR Control and how they are used to facilitate application-building and code integration.

I. MATFOR WIDGET

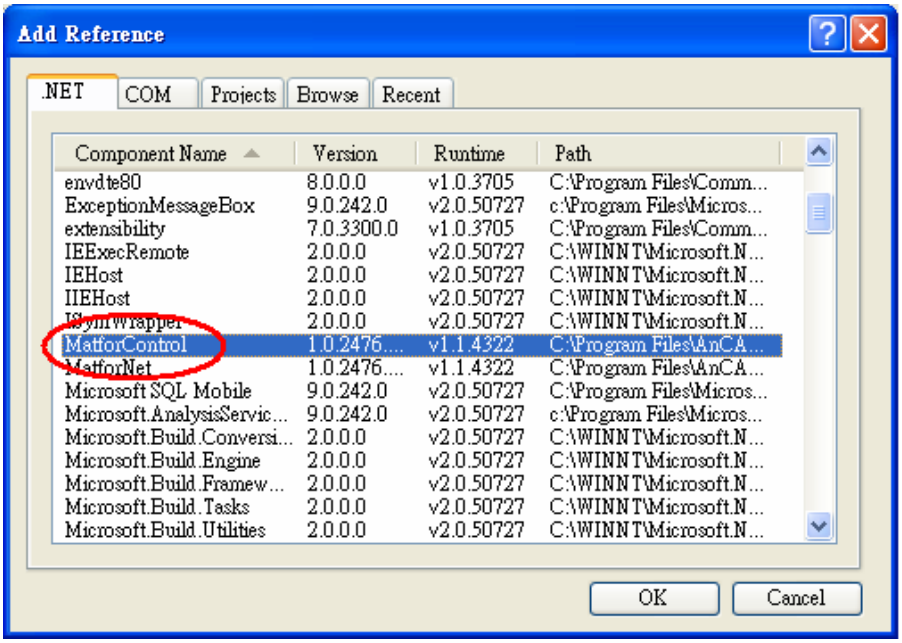
As a graphic component of MATFOR Control, MATFOR Widget component can be embedded into numerous UI design environments to enable application execution.

Integration with Microsoft Visual C# (ch5-1)

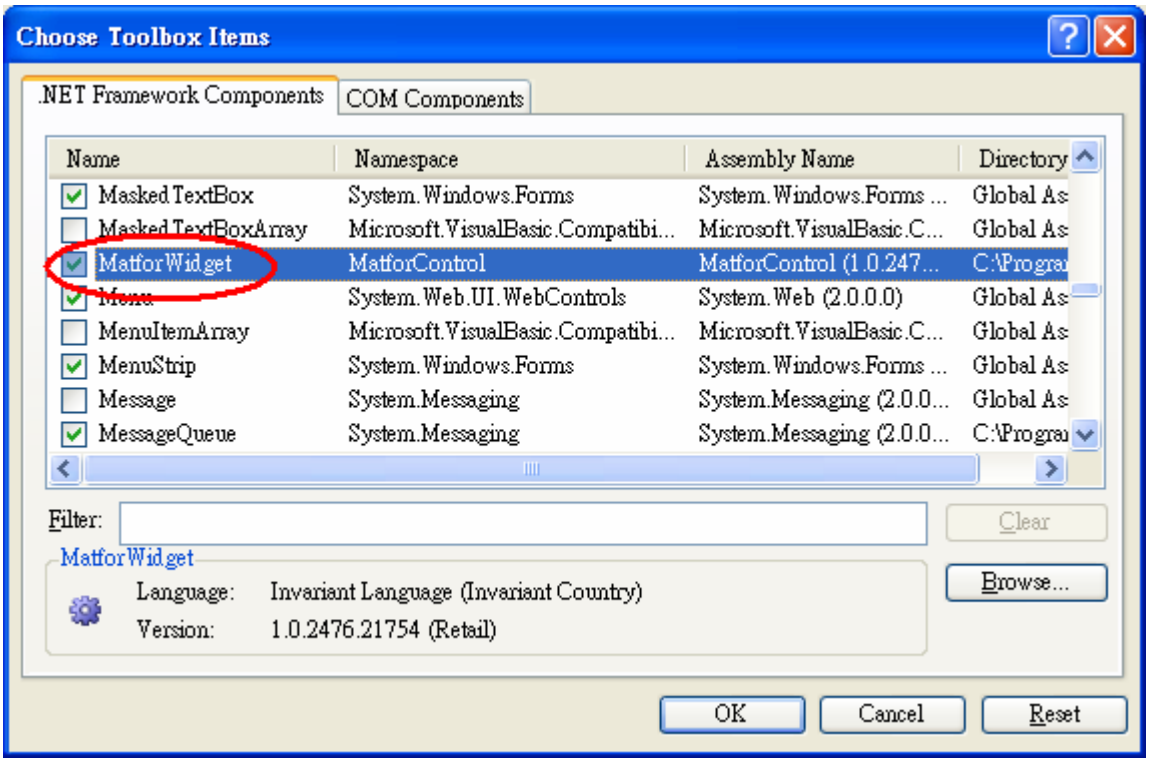
1. Select **Window Application** from **Templates**.
2. Name the project "**demo3**," specify its location, and click **OK**.



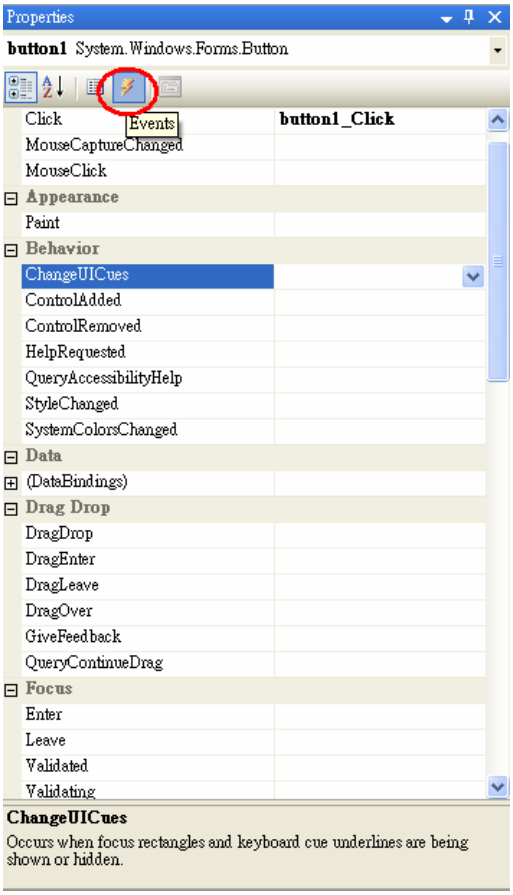
3. Select **Add Reference** from the **Project** menu.
4. Select **MatforNet** from the Component Name menu under the **.NET** tab and click **OK**.
5. Select **MatforControl** from the Component Name menu under the **.NET** tab and click **OK**.



6. On the **Tools** menu, click **Choose Toolbox Items**.
7. Under the **.NET Framework Components** tab, check the box next to **MatforWidget** and click **OK**. MatforWidget shall appear in Toolbox. (By default, the new item will be appended to the bottom of the list under current tab.)



8. Go to **Form1.cs [Design]**, add **Button** and **MatforWidget** to **Form1**.
9. Click on **button1**. Enter **Close** for Text under **Properties Window**.
10. Click the **Events** button in **Properties Window** and set **Click** to **button1_Click**.



11. Add the following codes in Form1.cs:

```
private void button1_Click(object sender, EventArgs e)
```

```
        {
            this.Close();
        }
```

12. Set **Load** to **Form1_Load**.
13. Add the following codes in Form1.cs::

```
        private void Form1_Load(object sender, EventArgs e)
        {
            mfArray x, y, z;
            mf.CreateSurfData(mf.Out(out x, out y, out z), 1, 30, 35);
            mf.Surf(x, y, z);
            mf.DrawNow();
        }
```

14. Add the following header to "**Form1.cs**":

```
    using MatforNet;
```

Compile and run