



Microsoft Excel から IMSL C# ライブラリを 使用方法

目次

背景.....	1
バックエンド	1
フロントエンド	3
複数のパラメータの受け渡し	4
配列の受け渡し	4
配列の返却.....	6
2次元 配列の計算	7
サマリー	8

背景

Microsoft 社は、Excel から .NET アプリケーションヘデータを取り込む API とドキュメントは提供しています。しかし、弊社のユーザは、Excel から IMSL の機能を使用したいと考えていらっしゃいます。 .NET の環境が更に進化し、Microsoft Office との統合性が向上すれば、Excel から IMSL の機能呼び出すことは容易になりますが、現在のところ、COM (共通オブジェクトモデル) が必要になります。IMSL C# ライブラリと Microsoft Excel のリンクに COM オブジェクトを使用する為、いくつかの制約があります。

まず第一に、C# 言語ではサポートされていない ComClass 属性を使用しなければなりません。従って、IMSL オブジェクトのインスタンスを作成するバックエンドの言語に Visual Basic を使用する必要があります。コマンドラインツールなどで、例題を作成することも可能ですが、ここで取り扱う例題は、Visual Studio.NET を使って作成します。

バックエンド

本セクションでは、ラッパーコード (Visual Basic.NET) を作成し、COM にアセンブリを登録する例を示します。

まず最初に、Visual Studio.NET で Visual Basic.NET のプロジェクトを新規作成します。クラスライブラリ テンプレートを使用し、COM に登録するための DLL を作成します。今回の例題では、プロジェクト名を ExcelTest とし、メインのソースファイル名はデフォルトの Class1.vb とします。

このライブラリの基本的なテンプレートは、次の通りです。

Namespace test

```
<ComClass(TestIMSL.ClassId, TestIMSL.InterfaceId, TestIMSL.EventsId)>_
Public Class TestIMSL
    Public Const ClassId As String = "6D0BC857-E796-4b39-BCB2-316ED4AF7F37"
    Public Const InterfaceId As String = "3B5C6011-E0F9-4cb5-A23D-4628A7388FE9"
    Public Const EventsId As String = "CB080278-0452-49a9-927A-D2F946BC2BC7"

    Public Sub New()
        MyBase.New()
    End Sub
End Class
End Namespace
```

MSDN のドキュメントに従いライブラリを正確に登録するためには、空のコンストラクタが必要になります。16 バイト (128 ビット) の 16 進数の ID 文字列は、GUIDs (Globally Unique Identifiers) と呼ばれ、ライブラリが binary confusion を避けるために一意的でなければなりません。

GUID を作成するには、IDE のツールメニューから GUID の作成を選択し、4 番目のフォーマットである Registry Format を使用します。Copy ボタンを押して、ソースコード ウィンドウに貼り付けるために作成した GUID をクリップボードにコピーします。貼り付けた GUID は中括弧 {} で囲まれているので、ダブルコーテーション " に変更します。ComClass 属性は、クラスの宣言と同一行でなければなりません。行を分割する場合は、行の継続を表す _ (スペースの後にアンダースコア) を使用します。

ここから先に進む前に、実際にコードを追加していきます。
まず、一つの倍精度の値を返却する 2 つのパブリック メソッドを作成します。一つのメソッドは、引数を持たず、もう一方のメソッドは、倍精度の値を引数として持ちます。ここで、Random オブジェクトのコンストラクタにシードを設定しないでください。毎回同じ乱数が生成されてしまいます。
IMSL ライブラリを使用するので、メニューバーのプロジェクトから参照の追加を選択し、IMSL C# Numerical Library の参照を追加します。
現在までの Visual Basic.NET のコードは以下の通りです。

Namespace test

```
<ComClass(TestIMSL.ClassId, TestIMSL.InterfaceId, TestIMSL.EventsId)>_  
Public Class TestIMSL  
    Public Const ClassId As String = "6D0BC857-E796-4b39-BCB2-316ED4AF7F37"  
    Public Const InterfaceId As String = "3B5C6011-E0F9-4cb5-A23D-4628A7388FE9"  
    Public Const EventsId As String = "CB080278-0452-49a9-927A-D2F946BC2BC7"  
  
    Dim R As New Imsl.Stat.Random  
  
    Public Function NextNormal() As Double  
        Return R.NextNormal()  
    End Function  
  
    Public Function Erf(ByVal dIn As Double) As Double  
        Return Sfun.Erf(dIn)  
    End Function  
  
    Public Sub New()  
        MyBase.New()  
    End Sub  
End Class  
End Namespace
```

次に、ライブラリのビルド プロパティの設定を行います。メニューバーのプロジェクトから、プロジェクトのプロパティを選択します。共通プロパティの全般にある出力の種類がクラスライブラリに設定されていることを確認します。次に構成プロパティのビルドを選択し、「COM の相互運用機能に登録」をチェックします。Visual Studio.NET を使用せず、コマンドライン ユーティリティを使ってアセンブリを COM に登録するには、次のコマンドを実行します。

```
regasm /tlb ExcelTest.dll
```

これは、COM に登録したバージョンを手動で更新する際にも使用できます。コードを作成し、IDE のオプションが正確に設定されると、アセンブリをビルドする準備が完了です。メニューバーのビルド ソリューションのビルドを選択し、コンパイル終了後、今回作成したプロジェクトの %bin ディレクトリに、新たに ExcelTest.dll が作成されます。Visual Studio.NET の出力ウィンドウに " 以前のプロジェクト出力の COM の相互運用機能への登録を解除しています..." と表示され、ウィンドウの下部に " プロジェクトの出力を COM の相互運用機能に登録しています..." と表示されます。つまり、ソリューションをビルドする度に、DLL が自動的に COM に登録されます。

これで、ライブラリが出来上がり、システムに登録されました。次に Excel 側の説明に移ります。

フロントエンド

新規に Excel のワークブックを作成し、Visual Basic Editor を開きます（ツール マクロ Visual Basic Editor もしくは Alt-F11）。最初に、Visual Basic Editor から先ほど新たに登録したアセンブリ（ExcelTest）を追加します。アセンブリを追加する方法は、ツール 参照設定を選択し、表示されるリストの中から登録してあるライブラリを示すチェックボックスをチェックし、OK をクリックします。それでは、これから新規モジュールを作成し（インサート モジュール）、バックエンドに作成したメソッドを呼び出す関数を作成します。

```
Public Function GetRand() As Double
    Dim o As New ExcelTest.TestIMSL
    GetRand = o.NextNormal()
End Function
```

```
Public Function GetErf(dbl) As Double
    Dim o As New ExcelTest.TestIMSL
    GetErf = o.Erf(dbl)
End Function
```

これらの関数を作成する際、Dim 行を打ち込んでいくと、ExcelTest の参照と TestIMSL クラスが自動的に表示されます。その後、“o.” と入力すると、利用可能なメソッドが表示されます。これらのダイアログにより、dll が正常にビルドされ、登録され、参照として追加されたことが確認できます。ワークブックを保存し、通常の Excel 画面に戻ります。GetRand() メソッドを使用するには、セル A1 を選択し関数として

```
=GetRand()
```

と入力します。Enter キーを押すと、以下のような乱数が表示されます。

```
0.800388
```

GetErf() メソッドを使用するには、例えばセル B1 に 0.5 を入力し、セル B2 に関数として

```
=GetErf(B1)
```

と入力します。Enter キーを押すと、次の様の計算結果が表示されます。

```
0.5205
```

アセンブリを呼び出すマクロ含んでいる Excel を開いたままで、Visual Basic.NET のコードを修正しようとしても、ソリューションをリビルドできません。Visual Studio は、“COM の相互運用機能への登録が失敗しました。アクセスが拒否されました。” などのエラーを出力します。バックエンドを修正するには、まず Excel を閉じる必要があります。

複数のパラメータの受け渡し

ここからは、複数の引数を使用する例題を示します。例えば、Imsl.Stat.Cdf にある Chi メソッドを考えます。このメソッドは、2つの入力パラメータを受け取り、1つの値を返却します。この場合、VB.NET 側では、メソッドは次のようになります。

```
Public Function ChiCdf(ByVal xIn As Double, ByVal nIn As Double) As Double
    Return Cdf.Chi(xIn, nIn)
End Function
```

Excel 側では、マクロで2つの引数を受け取ります。

```
Public Function GetChi(x, n) As Double
    Dim o As New ImslTest.ImslTestVB
    GetChi = o.ChiCdf(x, n)
End Function
```

Excel 表で使用する関数も、2つの値を渡す必要があるので、セル C1 に 0.15、セル C2 に 2.0 を設定し、セル C3 に、

```
=GetChi(C1,C2)
```

と入力します。
次のような計算結果が表示されます。

```
0.072257
```

配列の受け渡し

IMSL C# ライブラリを使って解析を行うために配列の値を VB.NET に渡す例をここに示します。例えば、VB.NET のコードは次のようになります。

```
Public Function Sign(ByRef xIn() As Double) As Double
    Dim st As SignTest
    st = New SignTest(xIn)
    Return st.Compute
End Function
```

ここでの入力パラメータは、倍精度の配列になります。倍精度のスカラーを渡すので、値渡し (ByVal) ではなく、参照渡し (ByRef) を使用することに注意してください。この例題では、VBA Excel 関数は、少し異なります。関数を入力する代わりに、配列または、範囲内のセルの値を渡します。ボタンコントロールを追加し、渡すセルの範囲をコード内で直接指定する方法が、最も簡単にプログラムする方法です。Excel のツールバー上で右クリック リストからコントロール ツールボックスを選択し、コントロール ツールボックスのツールバーを表示します。コマンドボタンを選択し、Excel のシート上に新しいボタンを作成します。作成したボタンを右クリック コマンドボタン オブジェクトの編集を選択した後、オブジェクト名を RunSgin に変更します。次に、再びボタンを右クリックし、コードの表示を選択します。Visual Basic Editor が

起動し、新しいページ（今回は Module1 ではなく、Sheet1）が表示されます。ここで、以下のようなコールバック関数を作成します。

```
Private Sub RunSign_Click()  
    Dim a(18) As Double  
    Dim i As Integer  
    Dim v As Double  
    Dim o As New ExcelTest.TestIMSL  
  
    ' 配列のロード  
    For i = 0 To 18  
        a(i) = Cells(i + 1, 4)  
    Next i  
  
    ' IMSL ルーチンの呼び出し  
    v = o.Sign(a)  
    Range("D21").Select  
    ActiveCell.FormulaR1C1 = v  
End Sub
```

入力配列は、Cells() 関数を使用して指定されたセル D1:D19 の範囲です。Sign() から返却された値は、セル D21 に表示されます。デザインボックスモードを終了し、Excel のシートに戻ります。この例題をテストするには、次の値を D1 から D19 に入力します。

```
92  
139  
-6  
10  
81  
-11  
45  
-25  
-4  
22  
2  
41  
13  
8  
33  
45  
-33  
-45  
-12
```

RunSign ボタンをクリックすると、計算結果がセル D21 に表示されます。

```
0.179642
```

配列の返却

配列を渡すだけでなく、返却値として配列を受け取る方法を考えます。今回の例題では、 x が、 $x[0,20]$ で 21 の値を持ち、 a の値が 1 つある場合に、 $\text{Gamma}(x, a)$ を返却します。ラッパーコードは次の通りです。

```
Public Function Gamma(ByVal aIn As Double) As Double()
    Dim i As Double
    Dim g(20) As Double
    For i = 0 To 20
        g(i) = Cdf.Gamma(i, aIn)
    Next i
    Return g
End Function
```

Excel 側では、入力パラメータを渡して、配列を受け取る必要があります。RunGamma という名のコマンドボタンを使用します。入力値として、セル E1 が使用され、出力配列は、E3:E23 の範囲に出力されます。

```
Private Sub RunGamma_Click()
    Dim o As New ExcelTest.TestIMSL
    ReDim g(20) As Double
    Dim x As Double
    Dim i As Integer

    ' 入力値を取得し、IMSL を呼び出す
    Range("E1").Select
    x = ActiveCell.Value
    g = o.Gamma(x)

    ' 出力セルに結果をコピー
    For i = 0 To 20
        Cells(i + 3, 5) = g(i)
    Next i
End Sub
```

配列 g の定義には、連続的なメモリのアロケートに Dim ではなく ReDim が使われていることに注意してください。セル番号 E1 に 5 を入力すると、次のような出力結果を表示します。

```
0 0.00366
0.052653
0.184737
0.371163
0.559507
0.714943
0.827008
0.900368
0.945036
0.970747
0.984895
0.9924
0.99626
```

```
0.998195
0.999143
0.9996
0.999815
0.999916
0.999962
0.999983
```

2次元配列の計算

2次元配列の取り扱いもそれほど複雑ではありません。この例題は、配列を渡して、その配列の逆行列を計算し、2次元配列として Excel に返却します。

Visual Basic.NET のコードは次の通りです。

```
Public Function Inverse(ByRef aIn As Double(,)) As Double(,)
    Dim lu As LU
    lu = New LU(aIn)
    Return lu.Inverse
End Function
```

入力配列は、ByRef として渡されることに注意してください。Excel VBA マクロは次のコマンドボタンにより実行されます。

```
Private Sub FindInverse_Click()
    Dim o As New ImslTest.ImslTestVB
    Dim a(2, 2) As Double
    ReDim inv(2, 2) As Double
    Dim i As Integer
    Dim j As Integer

    ' 2D 配列の作成
    For i = 0 To 2
        For j = 0 To 2
            a(i, j) = Cells(j, i + 6)
        Next j
    Next i

    ' IMSL ルーチンの呼び出し
    inv = o.Inverse(a)
    For i = 0 To 2
        For j = 0 To 2
            Cells(j + 4, i + 6) = inv(i, j)
        Next j
    Next i
End Sub
```

ここで、F1:H3 の 3x3 の行列を読み込み、その逆行列を F5:H7 に書き出します。以下のように、表示されます。

```
1 3 3
1 3 4
1 4 3
```

```
7 -3 -3
-1 0 1
-1 1 0
```

サマリー

ここで紹介した例題は、非常に基礎的なものですが、アセンブリを作成し、そのアセンブリを COM に登録する際に必要とされる全てのステップを示しているため、Excel マクロからパブリックメソッドにアクセスすることができます。

以下に Excel から IMSL C# ライブラリを使用する際のサマリーを示します。

- Visual Basic のクラスライブラリ プロジェクトを作成
- ComClass 属性にユニークな GUID を含める
- アセンブリを IDE もしくは、regasm で COM に登録
- Excel のモジュールにアセンブリを参照として含める