COMSOL Multiphysics®とLabVIEWによる初歩のリアルタイム計測/制御シミュレーション

計測エンジニアリングシステム株式会社
技術部: 前川 正喜

PC計測の世界で既にメジャーなソフトウェアであるLabVIEWと、シミュレーションソフトウェアのCOMSOL Multiphysics®を使用した、リアルタイムでの計測/制御とシミュレーションを行うための初歩的な手順についての説明です。LabVIEWとCOMSOL Multiphysics®を使用することで、実機での計測とシミュレーションを同時に行い、モデルの効率的な検証を行うことができます。

サンプルプログラム

LabVIEWのサンプルプログラムおよびxlamファイルは以下にアップロードしています。
https://kesco.co.jp/download/lllv.zip

LabVIEWはExcel VBAを介して、LiveLink™ for Excel®と通信します。通信方法は一般的にはTCPですが、64 bit環境で正しく動作しない場合があるため、NI-VISAを採用しました。
ワークブックをExcelアドイン(拡張子xlam)形式で保存しています。Excelマクロ有効ブック(拡張子xlsm)ではないことに注意してください。これはxlsmファイルでLiveLink™ for Excel®を実行すると、他のアプリケーションが開こうとする際、二重に開くことになり期待通りの動作を行わないためです。

xlamファイルの設定

Microsoft Visual Basic for Applicationsの参照設定ではComsolCom xxおよびVISA-COM xx Type Library(xxはバージョンによります)を登録しています。

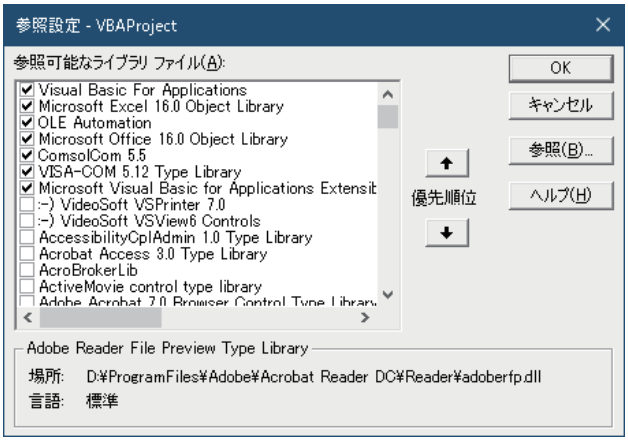

図1: 参照設定

コードウィンドウのプロシージャを記述する前の先頭行にModelUtilとComsolUtilクラスのオブジェクトを作成しています。また、NI-VISAのVisaComLibオブジェクトも作成します。

```
Dim ComsolUtil As ComsolCom.ComsolUtil
Dim ModelUtil As ComsolCom.ModelUtil
Dim ioMgr As VisaComLib.ResourceManager
Dim inst As VisaComLib.FormattedIO488
```

オブジェクトを格納するプロシージャには以下のように記述します。

```
Set ComsolUtil = CreateObject("comsolcom.
comsolutil")
Set ModelUtil = CreateObject("comsolcom.
modelutil")
Call ComsolUtil.TimeOuthandler(True)

Set ioMgr = New VisaComLib.ResourceManager
Set inst = New VisaComLib.FormattedIO488
Set inst.IO = ioMgr.Open("tcpip::127.0.0.1
::6666::socket")
inst.IO.TerminationCharacterEnabled = True
```

NI-VISAではポート番号`6666`、localhostの`127.0.0.1`を指定しています。LabVIEW側ではポート番号`6666`を指定して待機しているため、このsetステートメントが実行されるとTCP接続が確立されます。
また、終端文字を受信して読み取りを終了させるため、`TerminationCharacterEnabled`を`True`に設定しています。

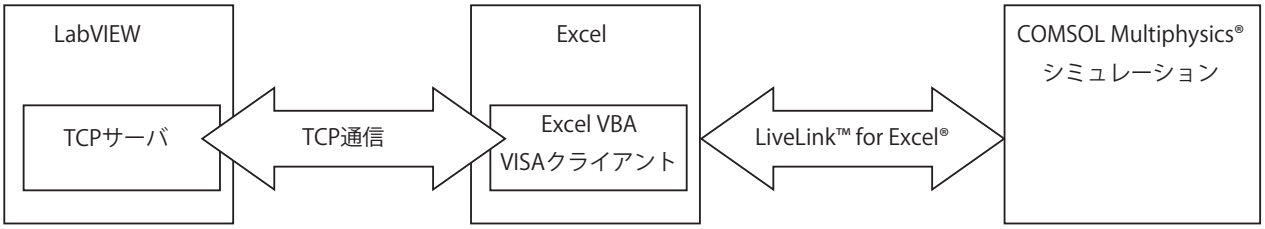

図12: LabVIEWとCOMSOL Multiphysics®シミュレーションの構成

LabVIEWの設定

LabVIEWがTCPサーバとなるため、クライアントが接続するまで待機します。

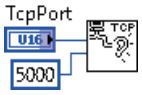

図2: TCP Listen.vi

TCPの書き込みは文末にLFを付加します。

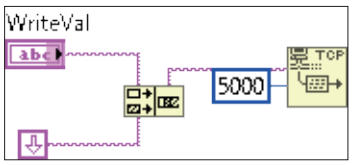

図3: TCP Write

xlamファイルを開くVIは以下の通りです。

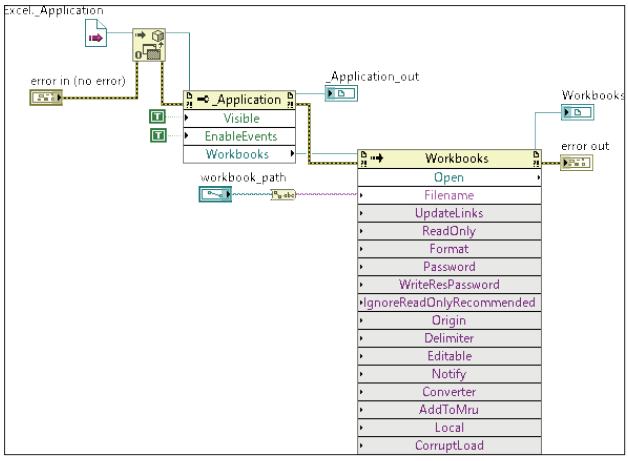

図4: xlamファイルを開く

プロシージャを指定するVIは、以下の通りです。

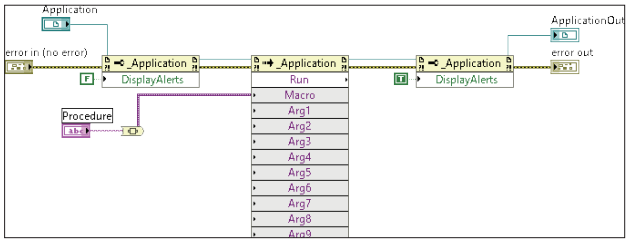

図5: プロシージャの指定

サンプルの実行

LiveLink™ for Excel®のサンプルファイルを開きます。使用するファイルは<COMSOLインストールフォルダ>Multiphysics\applications\LiveLink_for_Excel\Tutorialsにあるbusbar_llexcel.xlsxです。
次にCOMSOLメニューからbusbar_llexcel.mphを開きます。comsolmphserver.exeが起動しMPHファイルを読み込みます。読み込み中はLoading modelと表示されています。Loaded modelと表示されるまでお待ちください。

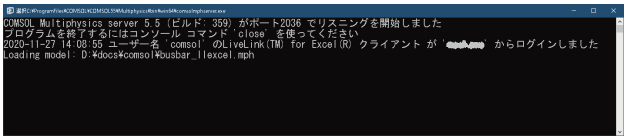

図6: comsolmphserver.exe

LabVIEWのサンプルを実行します。

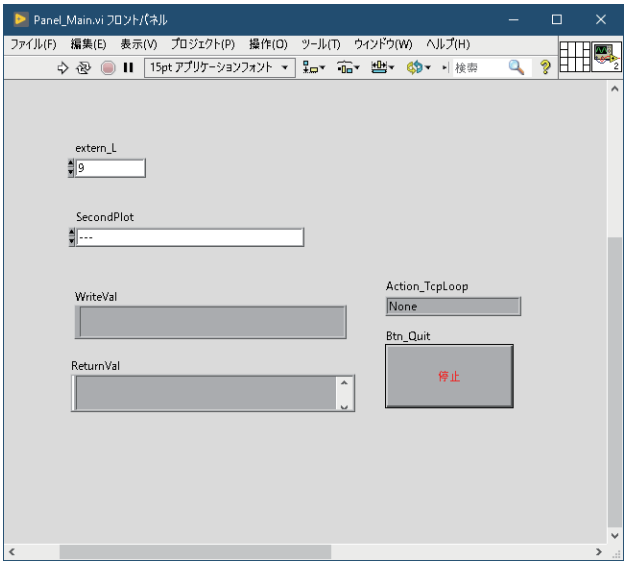

図7: LabVIEWのサンプルプログラム画面

実行するには➪をクリックします。
LabVIEWは以下の設定を逐次行います:

1. TCPでクライアントの接続待ち
2. xlamを開く

3. xlamファイルのプロシージャを実行

接続が確立されたら、数値制御器extern_Lの値を変更します。

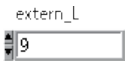

図8: extern_L

LabVIEWからxlamファイルにデータを送信します。
コマンドの書式は以下の通りです。

```
rCMD:rDATA(LF)
```

名前	説明
rCMD	コマンド部
:	コロン。セパレータ。
rDATA	値。
(LF)	改行。デリミタ

xlamファイルでは受信したデータをコマンド部と値に分けます。

```
Dim rBuf As String
Dim cNum As Long
cNum = InStr(rBuf, ":")
rCMD = Left(rBuf, cNum - 1)
rDATA = Mid(rBuf, cNum + 1)
rDATA = Replace(rDATA, vbLf, "")
```

次にコマンド部からプロシージャを呼び出します。

```
Select Case rCMD
   Case "extern_L"
       Call extern_L
   Case Else
End Select
```

値をbusbar_llexcel.xlsxのセルB5に入力し、スタディを実行します。

```
Range("b5").Value = rDATA
Set Model = ModelUtil.Model("Model")
Call ModelUtil.ShowProgress(True)
Call Model.get_study("std1").Run
```

進捗を表示(`ShowProgress(True)`)させている場合、進捗画面を表示します。

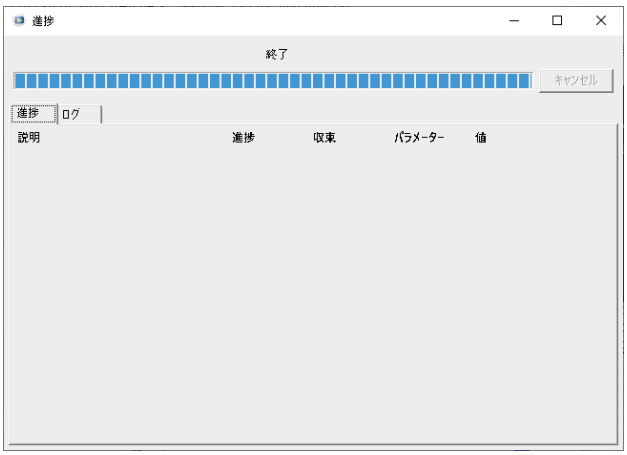

図9: 進捗画面

完了するとバスバーの長さが変更されます。
次にサブプロットを表示させます。
SecondPlotテキストリングから表示させたいプロットを選択します。

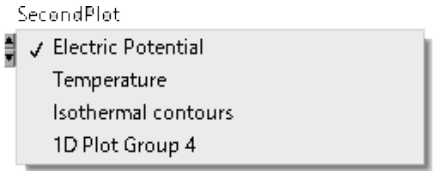

図10: SecondPlotテキストリング

選択するとLabVIEWからxlamファイルにデータを送信します。
xlamファイルでは受信した値から、プロットを表示させます。

```
Set Model = ModelUtil.Model("Model")
Dim PlotVal As String
PlotVal = "pg" & rDATA
Call Model.get_result(PlotVal).Run
```

プロットは最初のElectric Potentialを含め、4枚まで表示します。

最後に停止ボタンをクリックします。

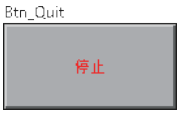

図11: 停止ボタン

アドインを開放し、プログラムを終了します。

xlamのソース

```
Dim ioMgr As VisaComLib.ResourceManager
Dim inst As VisaComLib.FormattedIO488
Dim rCMD As String
Dim rDATA As String
Dim SendData As String

Dim ComsolUtil As ComsolCom.ComsolUtil
Dim ModelUtil As ComsolCom.ModelUtil

Sub LLLV()
Call ComsoInit
Call VisaInit
End Sub

Sub ComsoInit()
Set ComsolUtil = CreateObject("comsolcom.
comsolutil")
Set ModelUtil = CreateObject("comsolcom.
modelutil")
Call ComsolUtil.TimeOuthandler(True)
Call ModelUtil.Connect
End Sub

Sub VisaInit()

Set ioMgr = New VisaComLib.ResourceManager
Set inst = New VisaComLib.FormattedIO488
Set inst.IO = ioMgr.Open("tcpip::127.0.0.1::666
6::socket")

inst.IO.TerminationCharacterEnabled = True

End Sub

Sub LLLVread()
Dim rBuf As String
Dim cNum As Long

inst.IO.Timeout = -1

On Error Resume Next
rBuf = inst.ReadString()
Err.Clear

cNum = InStr(rBuf, ":")
rCMD = Left(rBuf, cNum - 1)
rDATA = Mid(rBuf, cNum + 1)
rDATA = Replace(rDATA, vbLf, "")


Select Case rCMD

    Case "extern _ L"
        Call extern _ L
    Case "Plot"
        Call SecondPlot

    Case "QUIT"
        Call QuitLLLV

    Case Else
        '
End Select

End Sub

Sub LLLVwrite()

inst.WriteString SendData

End Sub

Sub extern _ L()

SendData = rDATA
Call LLLVwrite

Range("b5").Value = rDATA

Set Model = ModelUtil.Model("Model")
'para1 _ value = Sheets("Sheet1").Range("B5").
Value
Call ModelUtil.ShowProgress(True)
Call Model.get _ study("std1").Run

End Sub

Sub SecondPlot()

SendData = rDATA
Call LLLVwrite

Set Model = ModelUtil.Model("Model")
Dim PlotVal As String

PlotVal = "pg" & rDATA
Call Model.get _ result(PlotVal).Run

End Sub



Sub QuitLLLV()

SendData = rDATA
Call LLLVwrite

On Error Resume Next
Call Model.result().numerical().
Remove("interp")
Err.Clear
Call ModelUtil.Disconnect
Set ModelUtil = Nothing
Set ComsolUtil = Nothing


inst.IO.Close
Set inst = Nothing
Set ioMgr = Nothing

Dim ObjRef As Reference
Set ObjRef = ThisWorkbook.VBProject.
References("Module1")
Call ThisWorkbook.VBProject.References.
Remove(ObjRef)
On Error Resume Next
Call Workbooks("LLLV.xlam").Close(False)
On Error GoTo 0

End Sub
```