Murata Software

初めてのマクロセミナー

13：30－16：00
講師 伊勢 智之 @ムラタソフトウェア

muRata

目次

適当なタイミングで
休憩をはさみます

1.はじめに

【対象者】
当セミナーはVBAでのプログラム経験がない、Femtetのマクロ機能を使ったことがない、マクロ機能を使ったことはあるが、コードの内容がよくわからずカスタマイズすることができないといった方向けのセミナーです。

【セミナーの目標】
・Femtetのマクロ出力機能で出力されたコードの処理の流れをおおまかに理解する。

・実際にコードを変更してマクロを実行し動作確認を経て、マクロ編集のコツをつかむ。

1.はじめに

【マクロ機能を使うメリット】

メリット１　複雑なモデル作成の自動化

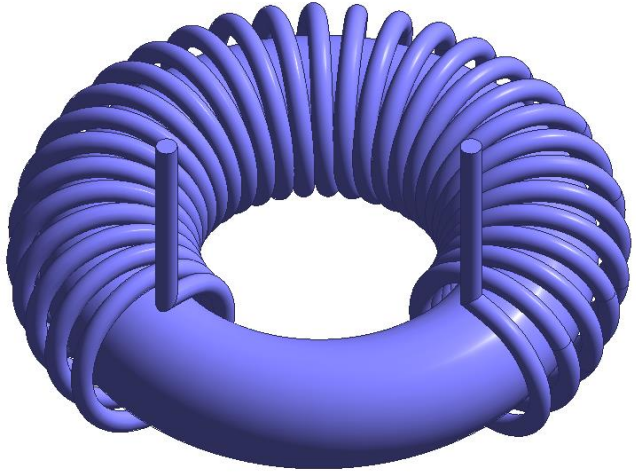
メリット２　解析結果分析の効率化

メリット３　ＣＡＥの普及、活用促進

メリット1 複雑なモデル作成の自動化

Femtetのモデル画面上からでは作成が困難なモデルや、手間のかかるモデルを自動で作成

	A	B	C	D	E	F	G	H	I	J
1	プロジェクト名	C:¥temp¥troic	(プロジェクトを保存するパスを入力します)							
2	トロイダルの小半径	4								
3	トロイダルの大半径	10								
4	トーラスの小半径	3								
5	コイルの巻数	20								
6	コイルの半径	0.5								
7	引出し線の長さ	10								
8	巻き終わり角度	270	(コイルの巻き始めの位置を0度として、巻き終わりの位置を角度で指定します。)							
9	コイルの多角柱化	1	(多角柱化する事で巻き数が多い場合のメッシュ生成エラーを回避します。							
10										
11	実行									
12										
13										

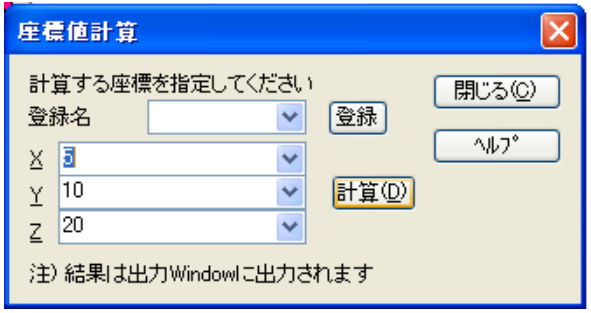


ムラタソフトウェアウェブサイトサンプルマクロページの事例2から
こちらのマクロをダウンロード可能。
https://www.muratasoftware.com/support/macro/

メリット2 解析結果分析の効率化

Murata Software

手間のかかる計算結果の取り出しを自動実行



取り出したい座標が多数ある場合、
データの取り出しが大変

マクロを使えばまとめてデータ取り出しが可能！

結果ファイル名(フ	C:¥temp¥ex1.pdt					
位置 座標X[mm]	位置 座標Y[mm]	位置 座標Z[mm]	計算実行	磁束密度 Bx	磁束密度 By	磁束密度 Bz
-10	-10	0				
-10	-5	0				
-10	0	0				
-10	5	0				
-10	10	0				
-5	-10	0				
-5	-5	0				
-5	0	0				
-5	5	0				
-5	10	0				
0	-10	0				
0	-5	0				

メリット3 ＣＡＥの普及、活用促進

開発部門

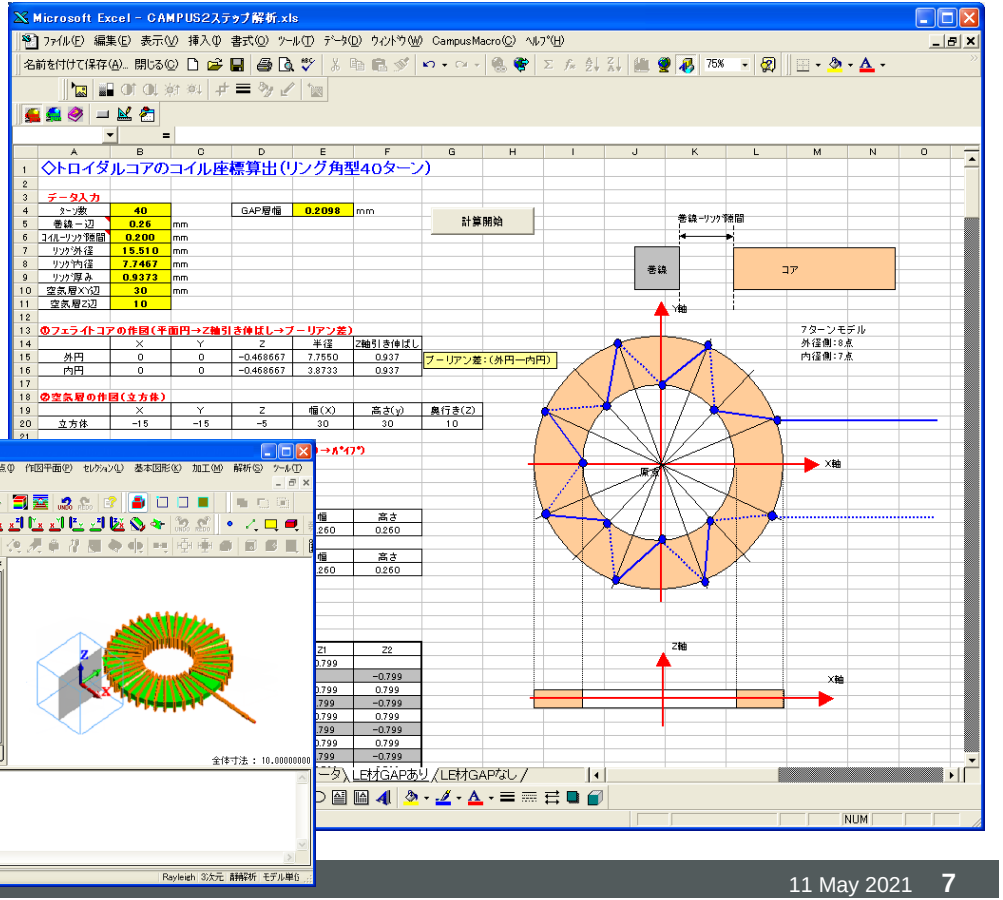
マクロで独自の解析システムを構築

（解析技術開発・要素開発）

↓

設計部門へ展開

Femtetを使ったことのないユーザーも
CAEの恩恵がうけられる



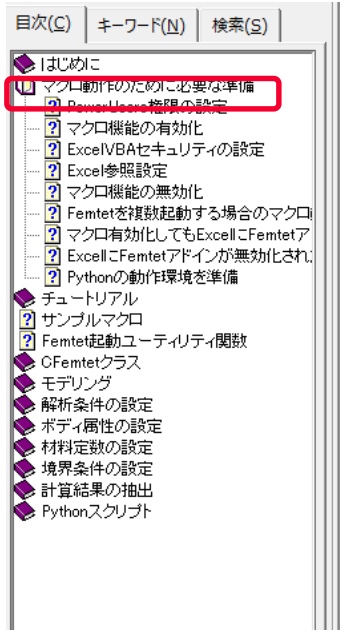
2.マクロを使用するための準備

マクロを使用するための準備事項についてご説明します。

2.マクロを使用するための準備

Murata Software

マクロ機能を使用するにはいくつか事前準備が必要です。
Femtetマクロヘルプの以下の項目を確認し、準備を完了してください。
「マクロ動作のために必要な準備」



マクロ動作のために必要な準備

マクロを動作させるには以下の準備作業が必要です。

手順

それぞれの詳細な手順はリンク先をご覧下さい。

手順	設定が必要なタイミング	設定可能ユーザ
1. PowerUsers権限の設定	1台のPCで初回の1回だけ	管理者(Administrator)権限を持つユーザ
2. マクロ機能の有効化	1台のPCで初回の1回だけ Femtetを再インストールした際、Officeのバージョンアップを行った際は、再度設定要	PowerUsers権限を持つユーザ(手順1で設定) Windows Vista / 7 の場合のみ、管理者(Administrator)権限を持つユーザ
3. ExcelVBAセキュリティの設定	各ログインユーザ毎	全ユーザ
4. Excel参照設定	Excelのファイル毎	全ユーザ

※参照設定だけはエクセルファイルごとに必要

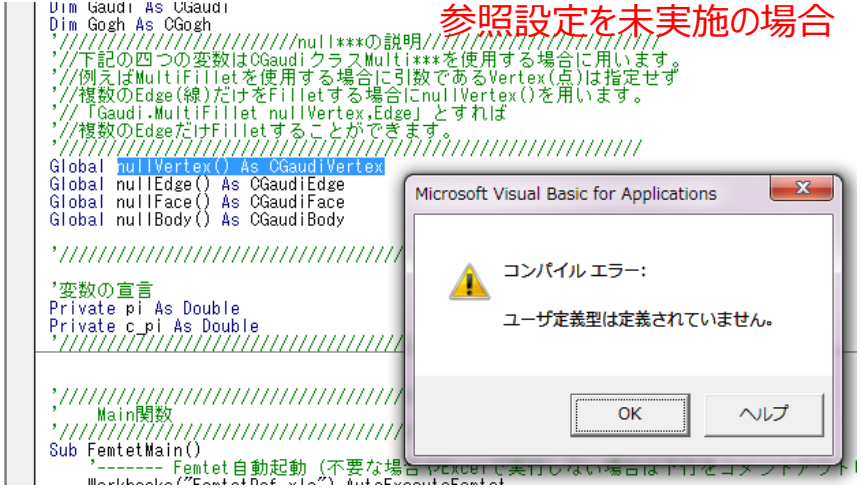
2.マクロを使用するための準備

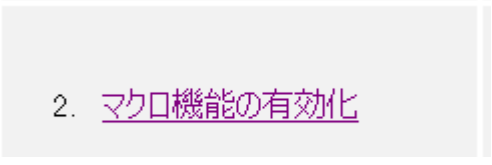
Murata Software

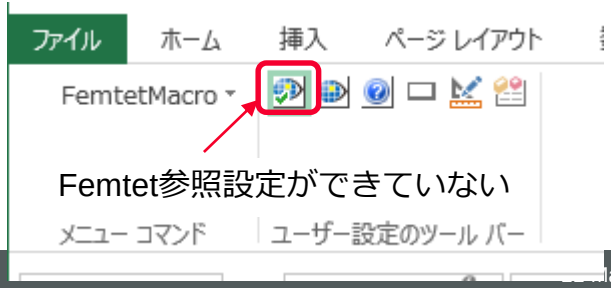
【準備が不十分の場合のエラー例】

有効化設定を未実施の場合

参照設定を未実施の場合







マクロ機能の有効化ができていない
⇒ マクロヘルプの「マクロ動作のために必要な準備」を確認

3.Femtetからマクロコードを出力し動作確認

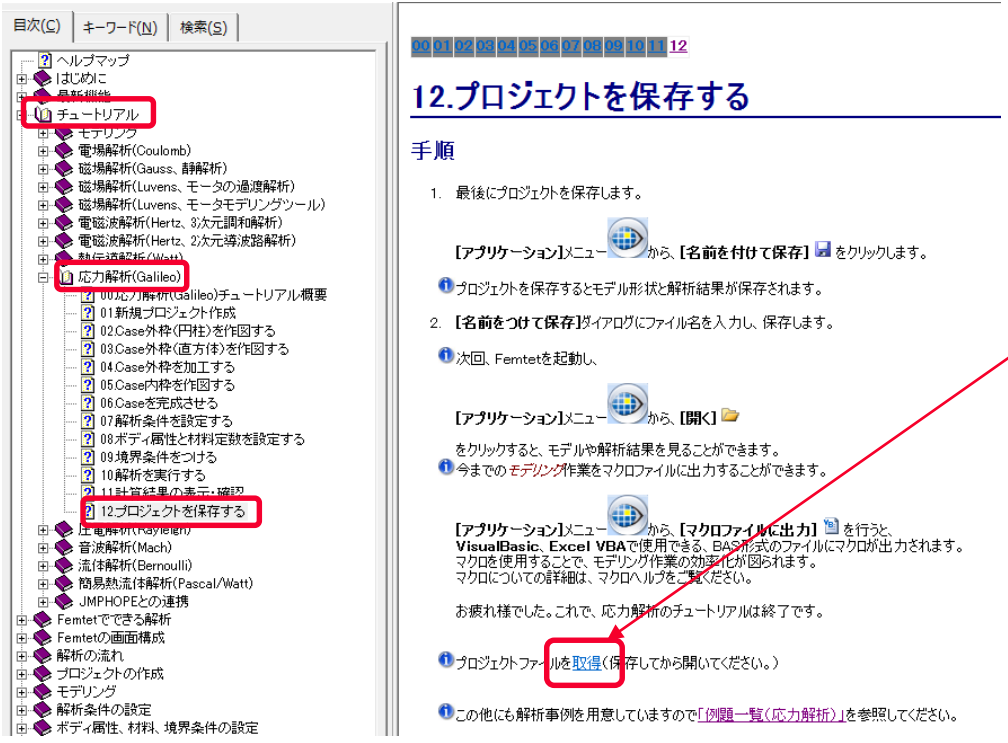
Murata Software

ではさっそくFemtetからマクロコードを出力し、
Excelで読み込んで動作させてみましょう。

3-1 プロジェクトファイルの取得
3-2 プロジェクトの動作確認
3-3 マクロファイルの出力
3-4 エクセルにマクロファイルを読み込む
3-5 マクロコードの確認
3-6 マクロを実行する
3-7 エラーが出た場合の対処
3-8.マクロ実行用ボタンの作成

3-1 プロジェクトファイルの取得

Femtetヘルプのチュートリアルからプロジェクトをダウンロードします。



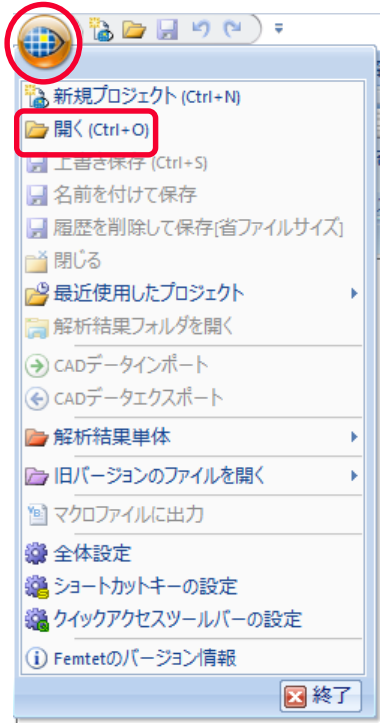
チュートリアル⇒応力解析(Galileo)⇒
12.プロジェクトを保存する のページを開く

「取得」のリンク部分をクリックし、
プロジェクトファイルを保存する

3-2 プロジェクトの動作確認

Murata Software

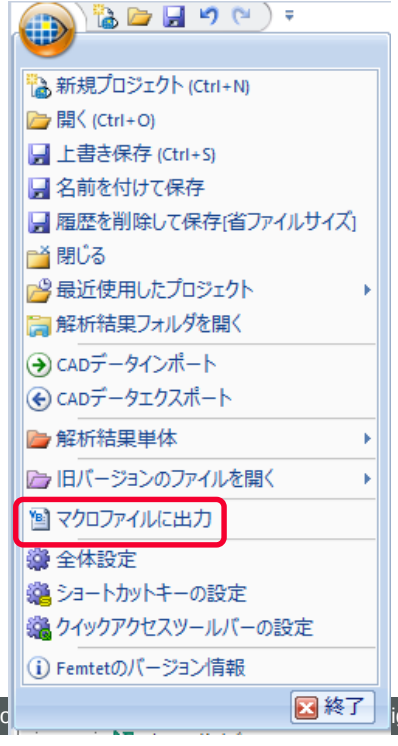
ダウンロードしたプロジェクトをFemtetで開きます。

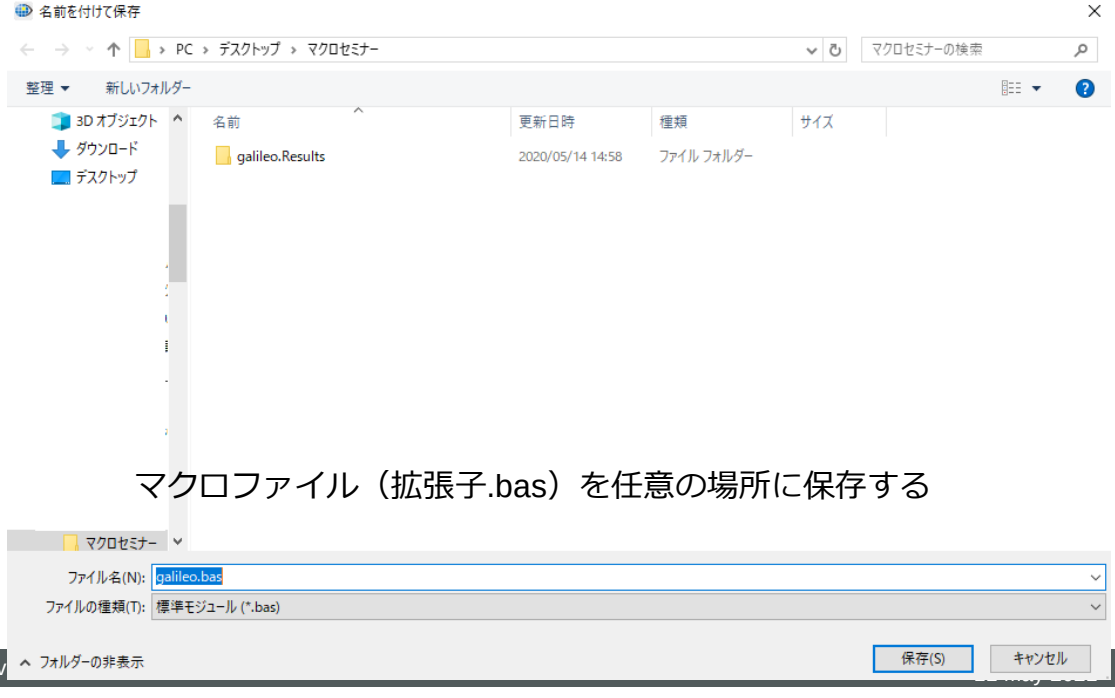


ようこそ画面の「プロジェクトを開く」または、
アプリケーションボタン⇒「開く」メニューを
実行し、ダウンロードしたプロジェクトを開く。

3-3 マクロファイルの出力

アプリケーションボタン⇒「マクロファイルに出力」を実行します。
モデル作成⇒条件設定⇒解析実行⇒結果抽出までのコードが生成されます。

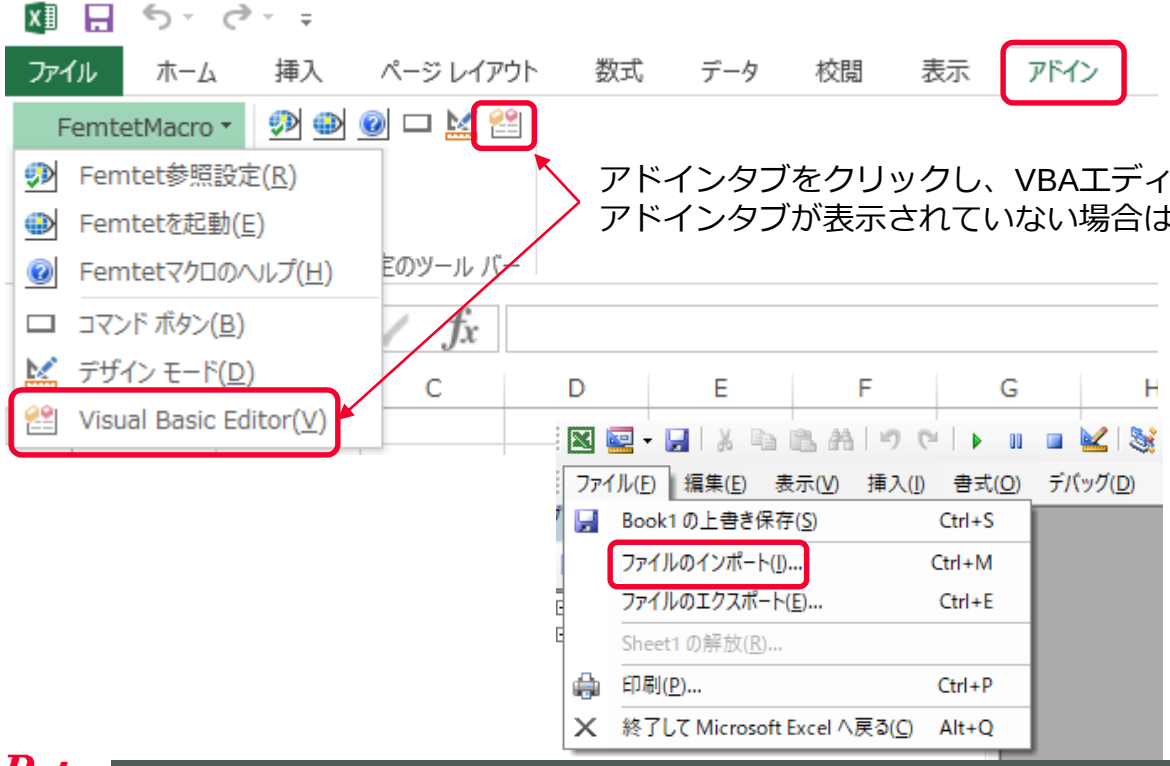




マクロファイル（拡張子.bas）を任意の場所に保存する

3-4 エクセルにマクロファイルを読み込む

Excelを起動し、エディタ（VBE）を開いてマクロファイルを読み込みます。



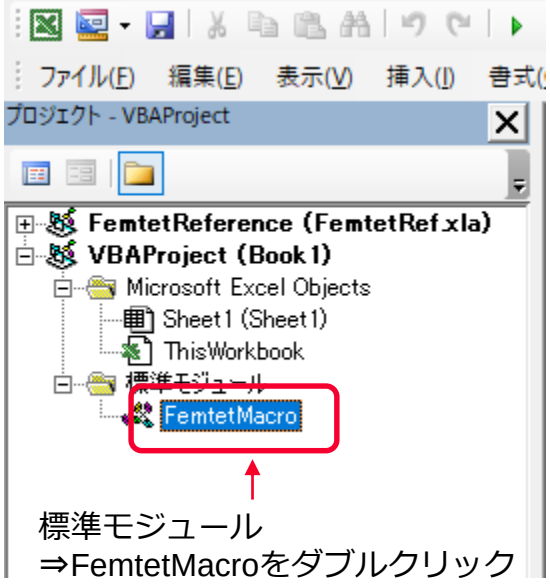
アドインタブをクリックし、VBAエディタを起動
アドインタブが表示されていない場合はマクロの準備ができていない可能性がある。

エディタ（VBE）が起動したら、
ファイルのインポートメニューを実行し
保存したマクロファイルをインポートする。

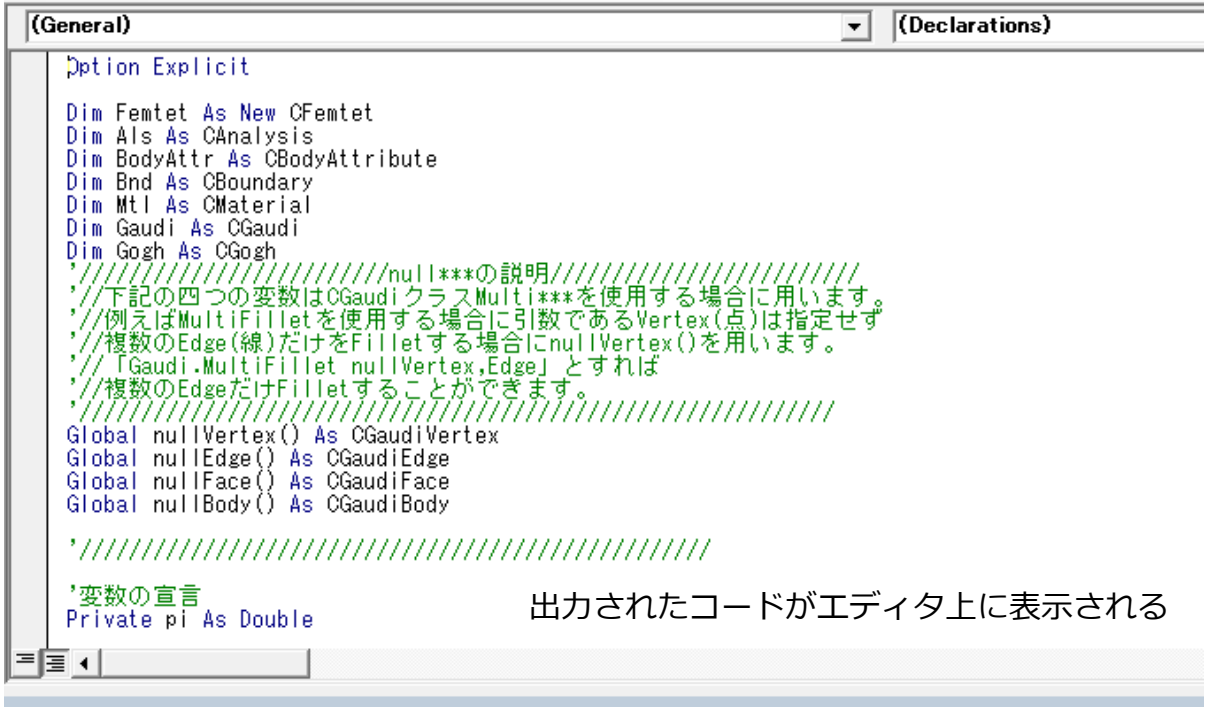
3-5 マクロコードの確認

Murata Software

インポートしたマクロコードを確認します。



標準モジュール
⇒FemtetMacroをダブルクリック

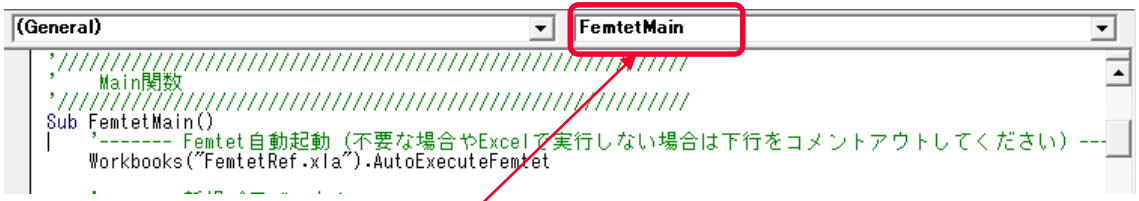


出力されたコードがエディタ上に表示される

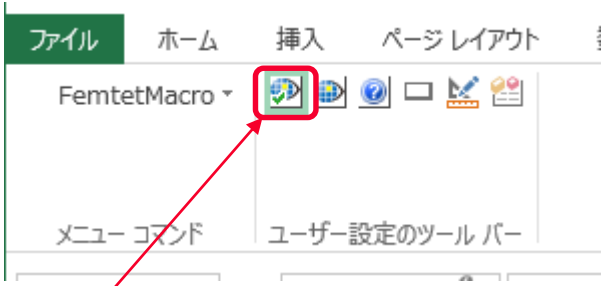
3-6 マクロを実行する

Murata Software

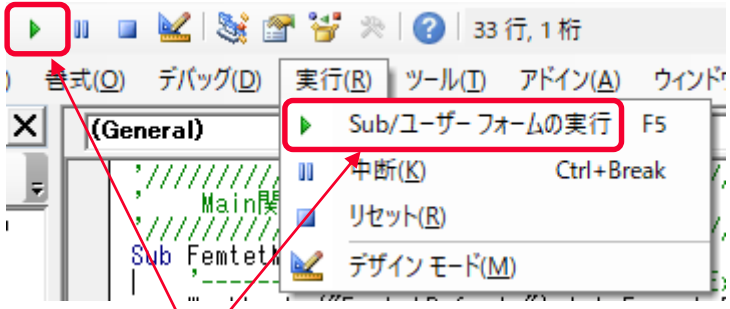
マクロを実行してみます。



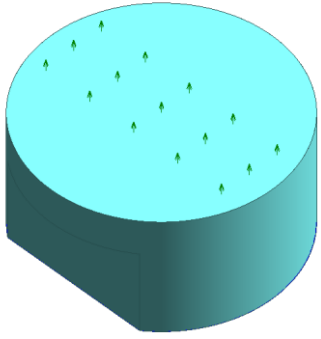
プロシージャのプルダウンメニューからFemtetMainを選択
⇒FemtetMain関数のコードに移動する



Excelのアドインタブ⇒Femtet参照設定を有効にする



「実行」タブ⇒「Sub/ユーザーフォームの実行」をクリック



モデルが自動生成される

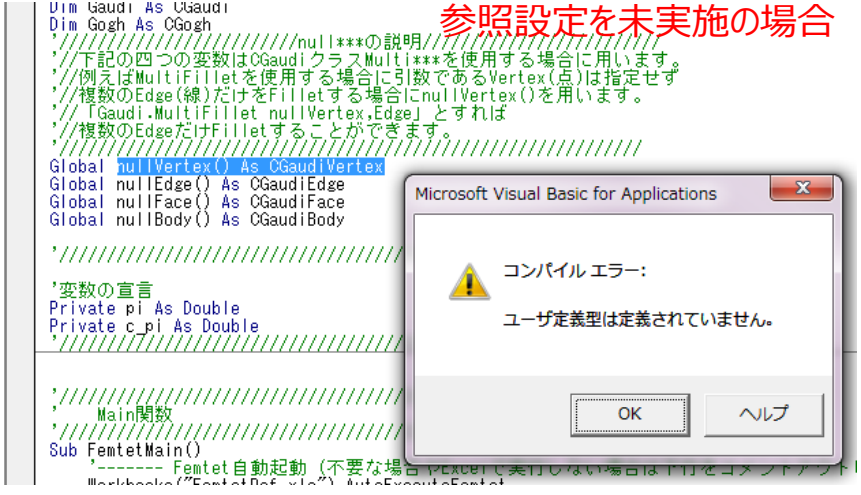
3-7 エラーが出た場合の対処

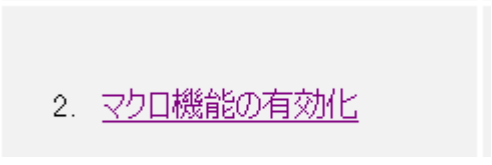
Murata Software

【マクロ実行時にエラーが出た場合】

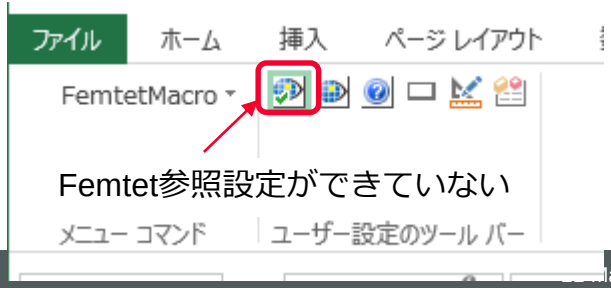
有効化設定を未実施の場合

参照設定を未実施の場合



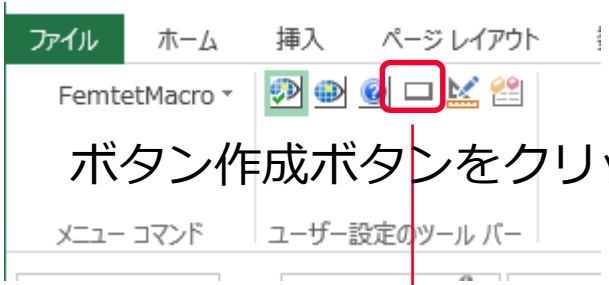


マクロ機能の有効化ができていない
⇒ マクロヘルプの「マクロ動作のために必要な準備」を確認

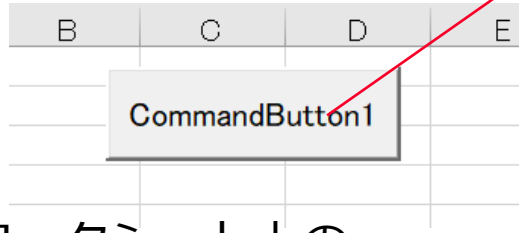


3-8.マクロ実行用ボタンの作成

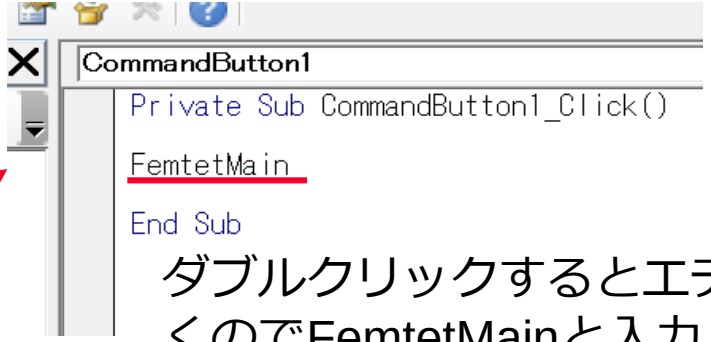
FemtetMain関数をボタンを押して実行できるようにします。



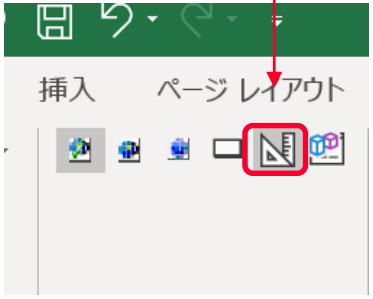
ボタン作成ボタンをクリック



ワークシート上の
適当な場所にボタンを作成
（マウスでドラッグ）



ダブルクリックするとエディターが開くのでFemtetMainと入力

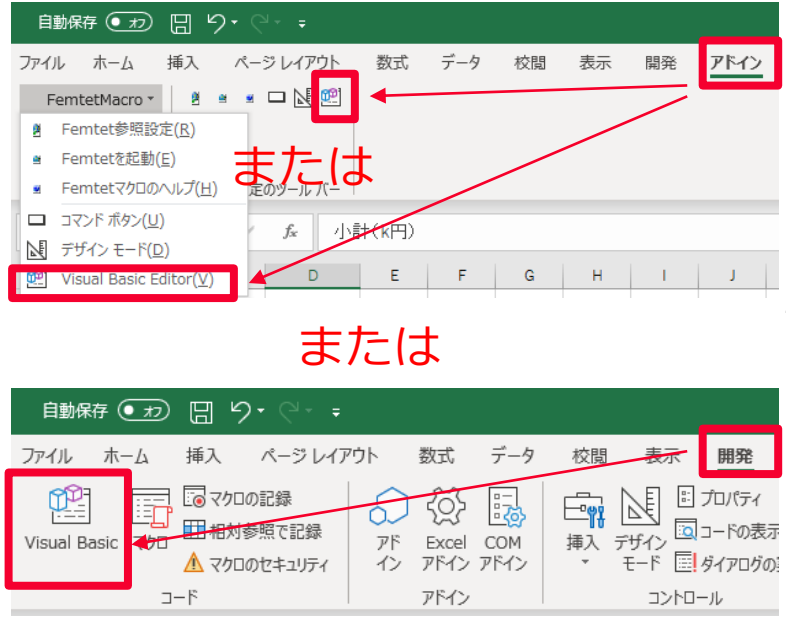


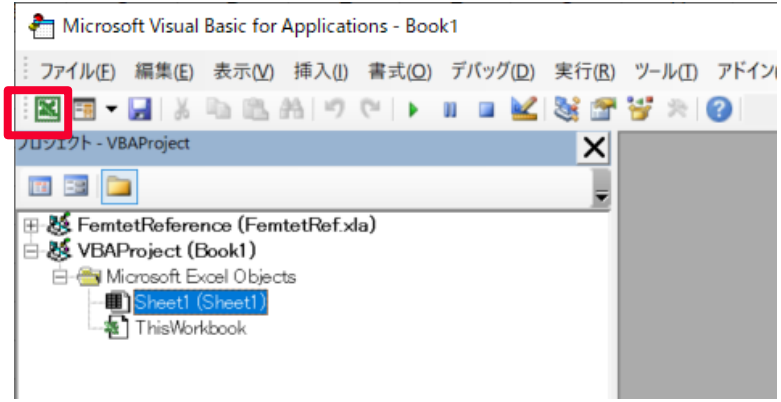
デザインモードボタンをクリックしてデザインモードから抜け、ボタンを押すとマクロが実行されることを確認します。

【補足】VBEとエクセルの画面切り替え

Murata Software

エクセル画面とVBE画面は簡単に切り替えることができます





エクセルからVBEを開く

VBEからエクセルに切り替える

【補足】マクロを含むエクセルファイル拡張子

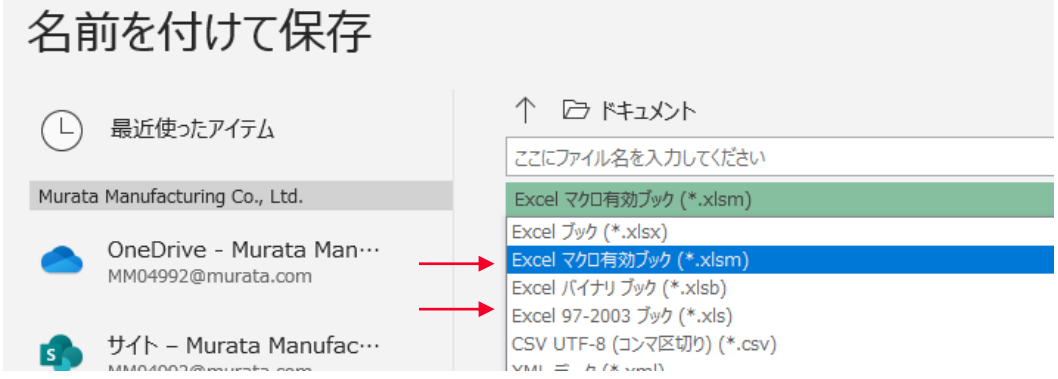
Murata Software

マクロコードを含むエクセルファイルの拡張子は以下のいずれかにします

〇〇〇.xls
〇〇〇.xlsm

※ 〇〇〇.xlsxではマクロを保存できません。

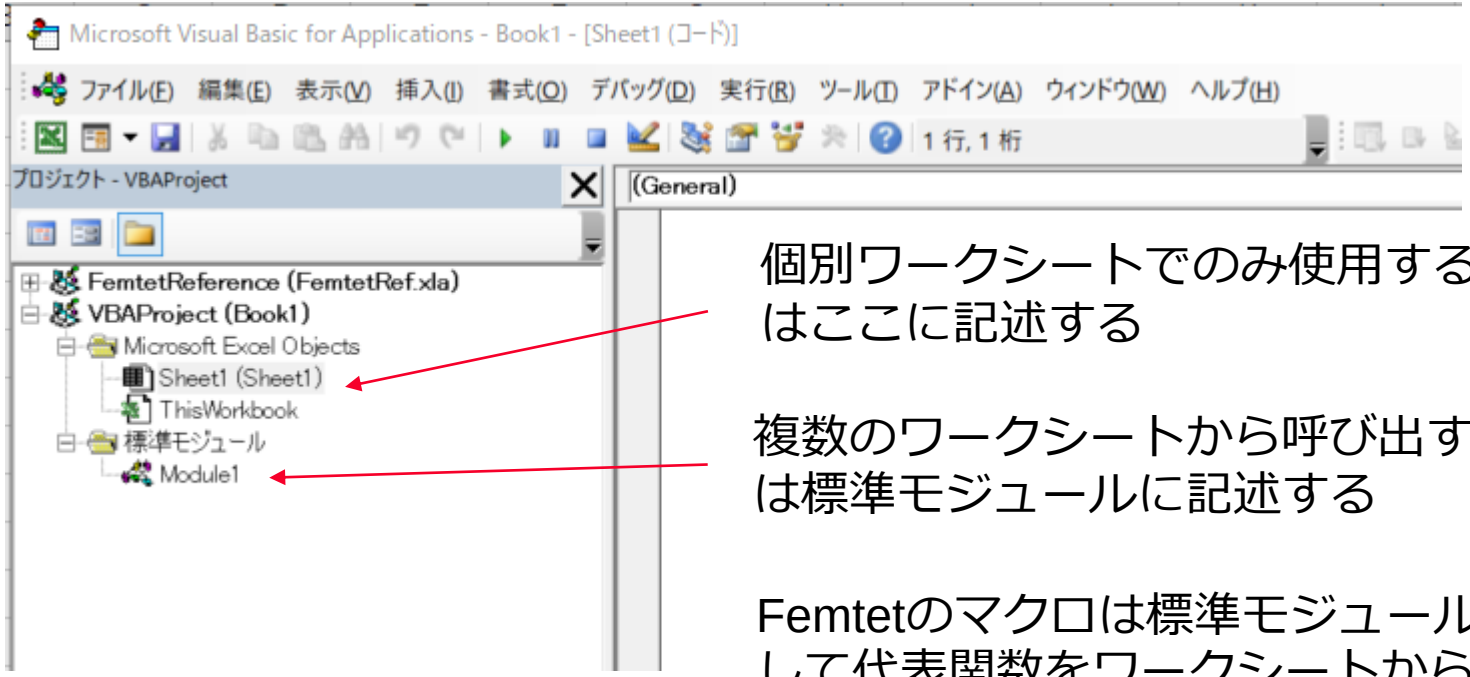
新規にエクセルファイルを作成し、マクロをインポートし編集した後に
名前を付けて保存する時の拡張子を*.xlsmまたは*.xlsとします。



【補足】マクロコードを書く場所について

Murata Software

各ワークシートまたは標準モジュールにコード（プログラム）を記述できる

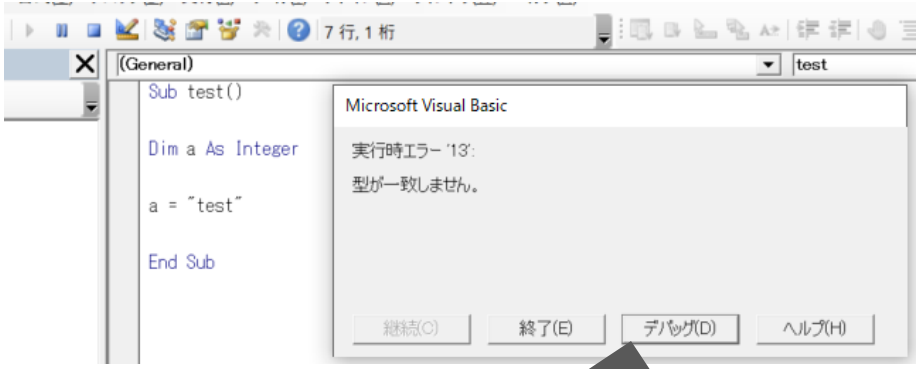


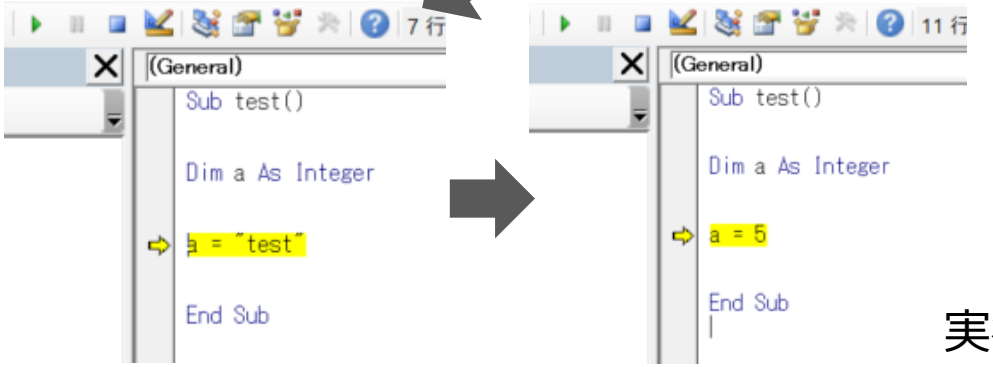
個別ワークシートでのみ使用するマクロはここに記述する

複数のワークシートから呼び出す共通関数は標準モジュールに記述する

Femtetのマクロは標準モジュールにインポートして代表関数をワークシートから呼び出すことができる

【補足】エラーとなった場合の対応



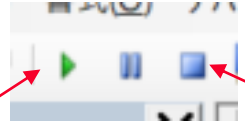


コードに不備がありエラーとなった場合、

デバッグボタンを押すとエラー箇所が黄色で表示される。

修正して処理を継続する場合は継続ボタンを押す。

初めから処理をやりなおす場合はリセットを押して実行ボタンを押す



実行/継続　　リセット

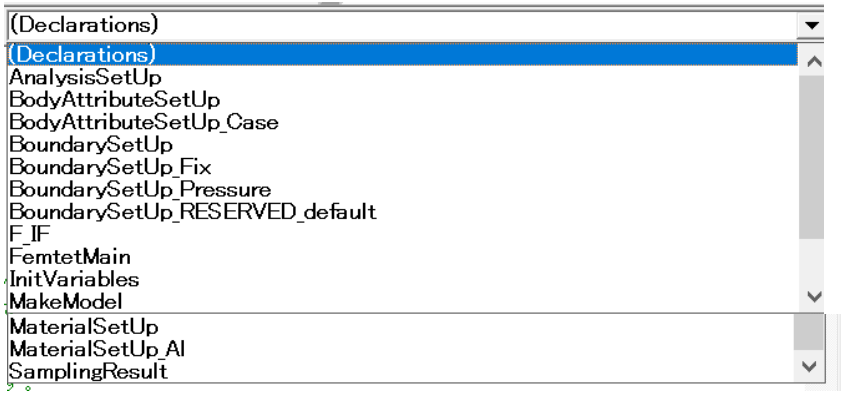
4. マクロコードの構成を理解する

Murata Software

4-1 プロシージャー（関数）のリスト
4-2 メイン関数の流れ
4-3 データベースの設定
4-4 モデル作成関数
4-5 結果取得関数
4-6 結果取得までマクロ実行する

4-1 プロシージャーのリスト

VBE（VisualBasicEditor)のプロシージャープルダウンメニューの▼ボタン
をクリックすると以下のようなプロシージャーのリストが見れます。



※順番はアルファベット順ですが、
記述の順番は異なります。

それぞれのプロシージャー（関数）
の名前から処理内容が予想できます

FemtetMain	メイン関数
AnalysisSetUp	解析条件設定
BodyAttributeSetUp	ボディ属性設定
BoundarySetUp	境界条件設定
MaterialSetUp	材料定数設定
MakeModel	モデル作成
SamplingResult	結果取得

4-2 メイン関数の流れ

Murata Software

```
'////////////////////////////////////////////////////////////
'     Main関数
'////////////////////////////////////////////////////////////
Sub FemtetMain()
    '------- Femtet自動起動 (不要な場合やExcelで実行しない場合は下行をコメントアウトしてください) --
    Workbooks("FemtetRef.xla").AutoExecuteFemtet

    '------- 新規プロジェクト -------
    If FEMTET.OpenNewProject() = False Then
        FEMTET.ShowLastError
    End If

    '------- 変数の定義 -------
    InitVariables

    '------- データベースの設定 -------
    AnalysisSetUp
    BodyAttributeSetUp
    MaterialSetUp
    BoundarySetUp

    '------- モデルの作成 -------
    Set Gaudi = FEMTET.Gaudi
    MakeModel

    '------- 標準メッシュサイズの設定 -------
    '<<<<<<< 自動計算に設定する場合は-1を設定してください >>>>>>>
    Gaudi.MeshSize = 2#

    '------- プロジェクトの保存 -------
    Dim ProjectFilePath As String
    ProjectFilePath = "C:¥Users¥MM04992¥Desktop¥SWCセミナー¥はじめてのマクロセミナー¥galileo"
    '<<<<<<< プロジェクトを保存する場合は以下のコメントを外してください >>>>>>>
    'If Femtet.SaveProject(ProjectFilePath & ".femprj", True) = False Then
    '    Femtet.ShowLastError
    'End If

    '------- メッシュの生成 -------
    '<<<<<<< メッシュを生成する場合は以下のコメントを外してください >>>>>>>
    'Gaudi.Mesh

    '------- 解析の実行 -------
    '<<<<<<< 解析を実行する場合は以下のコメントを外してください >>>>>>>
    'Femtet.Solve

    '------- 解析結果の抽出 -------
    '<<<<<<< 計算結果を抽出する場合は以下のコメントを外してください >>>>>>>
    'SamplingResult

End Sub
```

FemtetMain関数に全体の処理の流れが記述されています。

それぞれの処理内容はコメントからおおよそ以下の流れとなっています。

新規プロジェクト（作成）
データベース設定
モデル作成
メッシュサイズ設定
プロジェクト保存
メッシュ生成
解析
結果抽出

解析条件
ボディ属性
材料定数
境界条件

4-3 データベースの設定

Murata Software

解析条件設定	AnalysisSetUp	
ボディ属性設定	BodyAttributeSetUp	→ BodyAttributeSetUp_Case
材料定数設定	MaterialSetUp	→ MaterialSetUp_Al
境界条件設定	BoundarySetUp	→ BoundarySetUp_RESERVED_default BoundarySetUp_Fix BoundarySetUp_Pressure

解析条件以外は個別の属性の設定関数が呼ばれています。
当モデルはボディが一つなのでボディ属性、材料定数の個別関数は１つですが、
境界条件では３つの個別関数が呼ばれています。
個別関数の数はモデルによって変わります。

4-3 データベースの設定

Murata Software

AnalysisSetUpのコードと実際のFemtetの解析条件の設定を比較します

```
'////////////////////////////////////////////////////////////
'     解析条件の設定
'////////////////////////////////////////////////////////////
Sub AnalysisSetUp()

    '------- 変数にオブジェクトの設定 -------
    Set Als = FEMTET.Analysis

    '------- 解析条件共通(Common) -------
    Als.AnalysisType = GALILEO_C

    '------- 磁場(Gauss) -------
    Als.Gauss.b2ndEdgeElement = True

    '------- 電磁波(Hertz) -------
    Als.Hertz.b2ndEdgeElement = True

    '------- 調和解析(Harmonic) -------
    Als.Harmonic.FreqSweepType = LINEAR_INTERVAL_C

    '------- 熱荷重(ThermalStress) -------
    Als.ThermalStress.Set_Table 1, (0)

    '------- 高度な設定(HighLevel) -------
    Als.HighLevel.nNonL = (20)
    Als.HighLevel.bATS = False
    Als.HighLevel.FactorType = RADIO_ANALYTICAL_C
```

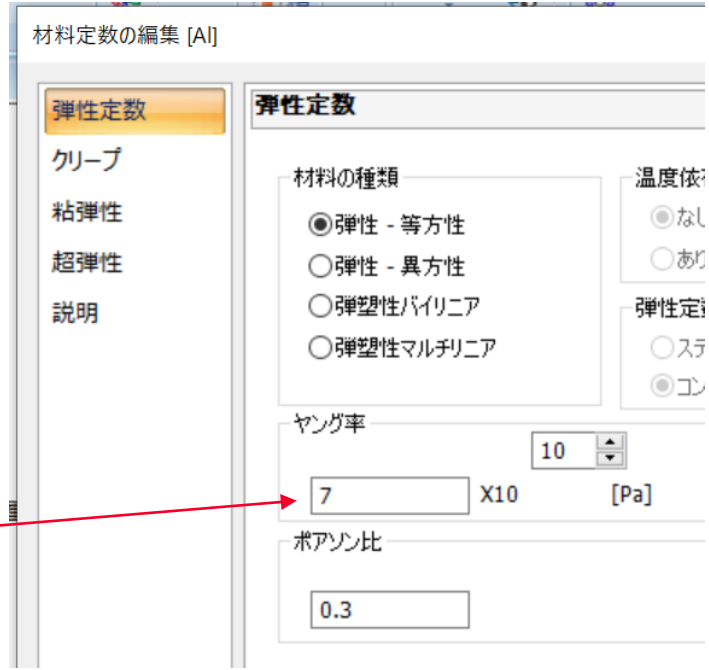
ソルバ選択や解析オプション設定など
各種解析条件が記述されます

4-3 データベースの設定

Murata Software

MaterialSetUp_Alのコードと実際のFemtetの材料Alの設定を比較します

```
'////////////////////////////////////////////////////////////
'    Materialの設定 Material名：Al
'////////////////////////////////////////////////////////////
Sub MaterialSetUp_Al()
    '------- MaterialのIndexを保存する変数 -------
    Dim Index As Integer

    '------- Materialの追加 -------
    Mtl.Add "Al"

    '------- Material Indexの設定 -------
    Index = Mtl.Ask("Al")

    '------- 線膨張係数(Expansion) -------
    Mtl.Expansion(Index).sAlf = (0#)
    Mtl.Expansion(Index).Set_vAlf 0, (0#)
    Mtl.Expansion(Index).Set_vAlf 1, (0#)
    Mtl.Expansion(Index).Set_vAlf 2, (0#)

    '------- 弾性定数(Elasticity) -------
    Mtl.Elasticity(Index).sY = (7) * 10 ^ (10)

    '------- 粘度(Viscosity) -------
    Mtl.Viscosity(Index).Mu = (1.002) * 10 ^ (-3)

    '------- 着磁(Magnetize) -------
    Mtl.Magnetize(Index).MagRatioType = MAGRATIO_BR_C
End Sub
```

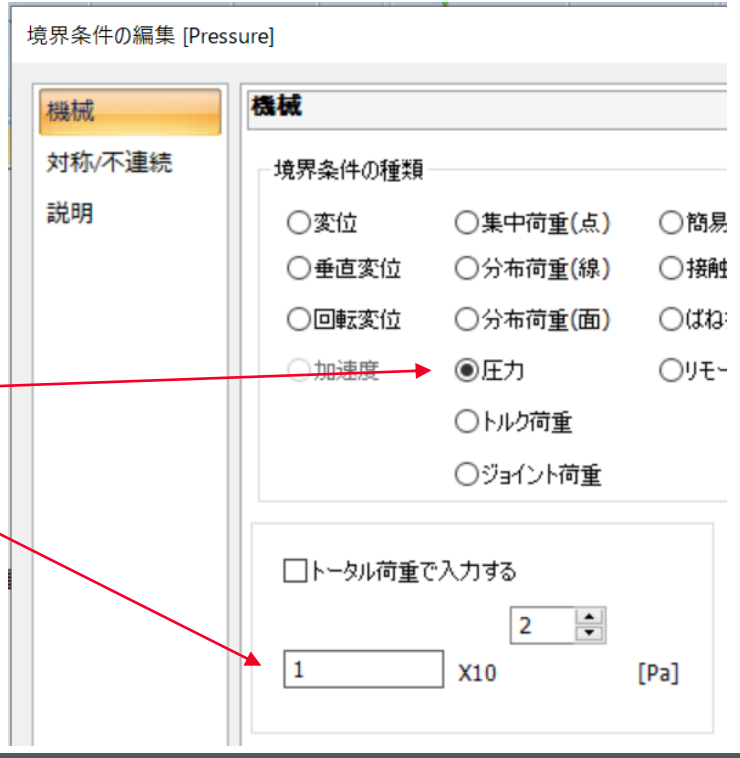


4-3 データベースの設定

Murata Software

BoundarySetUp_Pressureのコードと実際のFemtetの境界条件Pressureの設定を比較します

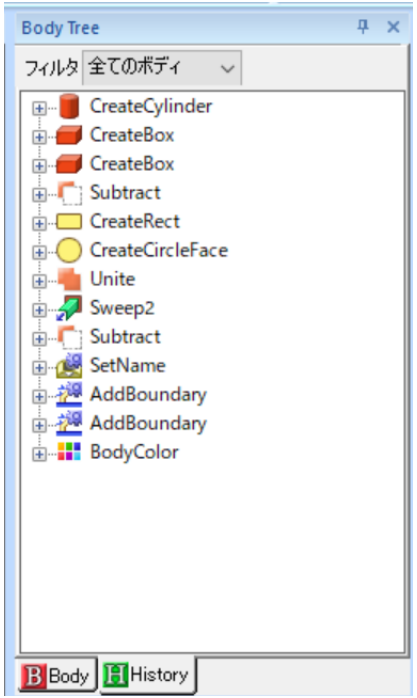
```
'////////////////////////////////////////////////////////////
'    Boundaryの設定 Boundary名：Pressure
'////////////////////////////////////////////////////////////
Sub BoundarySetUp_Pressure()
    '------- BoundaryのIndexを保存する変数 -------
    Dim Index As Integer

    '------- Boundaryの追加 -------
    Bnd.Add "Pressure"

    '------- Boundary Indexの設定 -------
    Index = Bnd.Ask("Pressure")

    '------- 機械(Mechanical) -------
    Bnd.Mechanical(Index).Condition = PRESSURE_C
    Bnd.Mechanical(Index).P = (1) * 10 ^ (2)

    '------- 室温_環境温度(RoomTemp) -------
    Bnd.Thermal(Index).RoomTemp.Temp = (0)

    '------- 流体(Fluid) -------
    Bnd.Fluid(Index).VelocityCondition = XYZ_VELOCITY_C

    '------- 流体(FluidBern) -------
    Bnd.FluidBern(Index).Condition = SOLID_WALL_C
    Bnd.FluidBern(Index).P = (0#)
    Bnd.FluidBern(Index).TempType = TEMP_DIRECT_C
End Sub
```



4-4 モデル作成関数（MakeModel）

```
'////////////////////////////////////////////////////////
'     モデル作成関数
'////////////////////////////////////////////////////////
Sub MakeModel()

    '------- Body配列変数の定義 -------
    Dim Body() As CGaudiBody

    '------- モデルを描画させない設定 -------
    FEMTET.RedrawMode = False


    '------- CreateCylinder -------
    ReDim Preserve Body(0)
    Dim Point0 As New CGaudiPoint
    Point0.SetCoord 0#, 0#, 0#
    Set Body(0) = Gaudi.CreateCylinder(Point0, 10#, 10#)

    '------- CreateBox -------
    ReDim Preserve Body(1)
    Dim Point1 As New CGaudiPoint
    Point1.SetCoord -10#, 8#, 0#
    Set Body(1) = Gaudi.CreateBox(Point1, 20#, 4#, 7#)

    '------- CreateBox -------
    ReDim Preserve Body(2)
    Dim Point2 As New CGaudiPoint
    Point2.SetCoord -10#, -12#, 0#
    Set Body(2) = Gaudi.CreateBox(Point2, 20#, 4#, 7#)

    '------- Subtract -------
    ReDim Preserve Body(3)
    Dim BodyArray0(1) As CGaudiBody
    Dim BodyArray1() As CGaudiBody
    Set BodyArray0(0) = Body(1)
    Set BodyArray0(1) = Body(2)
    Body(0).Subtract2 BodyArray0, BodyArray1, 1
    Set Body(3) = BodyArray1(0)

    '------- CreateRect -------
    ReDim Preserve Body(4)
    Dim Point3 As New CGaudiPoint
    Point3.SetCoord -8#, -4#, 0#
    Set Body(4) = Gaudi.CreateRect(Point3, 16#, 8#)
```

```
    '------- CreateCircleFace -------
    ReDim Preserve Body(5)
    Dim Point4 As New CGaudiPoint
    Point4.SetCoord 0#, 0#, 0#
    Set Body(5) = Gaudi.CreateCircleFace(Point4, 5#)

    '------- Unite -------
    ReDim Preserve Body(6)
    Dim BodyArray2(0) As CGaudiBody
    Dim BodyArray3() As CGaudiBody
    Set BodyArray2(0) = Body(5)
    Body(4).Unite BodyArray2, BodyArray3, True
    Set Body(6) = BodyArray3(0)

    '------- Sweep2 -------
    Dim Vector0 As New CGaudiVector
    Vector0.SetCoord 0#, 0#, 9#
    Body(6).Sweep2 Vector0

    '------- Subtract -------
    ReDim Preserve Body(7)
    Dim BodyArray4(0) As CGaudiBody
    Dim BodyArray5() As CGaudiBody
    Set BodyArray4(0) = Body(6)
    Body(3).Subtract2 BodyArray4, BodyArray5, 1
    Set Body(7) = BodyArray5(0)

    '------- SetName -------
    Body(7).SetName "Case", "Al"

    '------- AddBoundary -------
    Dim Face0 As CGaudiFace
    Set Face0 = Body(7).GetFaceByID(9)
    Face0.AddBoundary "Fix"

    '------- AddBoundary -------
    Dim Face1 As CGaudiFace
    Set Face1 = Body(7).GetFaceByID(325)
    Face1.AddBoundary "Pressure"

    '------- AddBoundary -------
    Dim Face1 As CGaudiFace
    Set Face1 = Body(7).GetFaceByID(325)
    Face1.AddBoundary "Pressure"

    '------- BodyColor -------
    Body(7).Color = &HFFFF80


    '------- モデルを再描画します -------
    FEMTET.Redraw

End Sub
```



MakeModel関数にはモデル作成の履歴が記述されています。
Historyタブの記録と対応しています。

4-5 結果取得関数

```
'////////////////////////////////////////////////////////
'     計算結果抽出関数
'////////////////////////////////////////////////////////
Sub SamplingResult()

    '------- 変数にオブジェクトの設定 -------
    Set Gogh = FEMTET.Gogh

    FEMTET.SavePDT FEMTET.ResultFilePath & ".pdt", True 'pdtファイルを保存します
    FEMTET.OpenPDT FEMTET.ResultFilePath & ".pdt", True 'pdtファイルを開きます

    '------- フィールドの設定 -------
    Gogh.Galileo.Vector = GALILEO_DISPLACEMENT_C

    '------- 最大値の取得 -------
    Dim PosMax() As Double '最大値の座標
    Dim ResultMax As Double '最大値

    If Gogh.Galileo.GetMAXVectorPoint(VEC_C, CMPX_REAL_C, PosMax, ResultMax) = False Then
        FEMTET.ShowLastError
    End If

    '------- 最小値の取得 -------
    Dim PosMin() As Double '最小値の座標
    Dim ResultMin As Double '最小値

    If Gogh.Galileo.GetMINVectorPoint(VEC_C, CMPX_REAL_C, PosMin, ResultMin) = False Then
        FEMTET.ShowLastError
    End If

    '------- 任意座標の計算結果の取得 -------
    Dim Value() As New CComplex

    If Gogh.Galileo.GetVectorAtPoint(0, 0, 0, Value()) = False Then
        FEMTET.ShowLastError
    End If

    ' 複数の座標の結果をまとめて取得する場合は、MultiGetVectorAtPoint関数をご利用ください。

End Sub
```

データベース設定やモデル作成
はFemtetの設定やモデル作成履歴が
そのままコードとして出力される。

しかし結果取得関数はソルバー設定に
応じた代表的な結果取得のサンプル
コードが出力されているにすぎない。

応力解析の場合は、最大変位、最小変位
任意座標の変位ベクトルを取得する
サンプルコードが記述されている。

カスタマイズするにはマクロヘルプを
参考にしてコードを編集する必要がある。
→難易度はやや高い

4-6 結果取得までマクロ実行する

Murata Software

```
'////////////////////////////////////////////////////////
'     計算結果抽出関数
'////////////////////////////////////////////////////////
Sub SamplingResult()

    '------- 変数にオブジェクトの設定 -------
    Set Gogh = FEMTET.Gogh

    FEMTET.SavePDT FEMTET.ResultFilePath & ".pdt", True 'pdtファイルを保存します
    FEMTET.OpenPDT FEMTET.ResultFilePath & ".pdt", True 'pdtファイルを開きます

    '------- フィールドの設定 -------
    Gogh.Galileo.Vector = GALILEO_DISPLACEMENT_C

    '------- 最大値の取得 -------
    Dim PosMax() As Double '最大値の座標
    Dim ResultMax As Double ' 最大値

    If Gogh.Galileo.GetMAXVectorPoint(VEC_C, CMPX_REAL_C, PosMax, ResultMax) = False Then
        FEMTET.ShowLastError
    End If

    '------- 最小値の取得 -------
    Dim PosMin() As Double '最小値の座標
    Dim ResultMin As Double '最小値

    If Gogh.Galileo.GetMINVectorPoint(VEC_C, CMPX_REAL_C, PosMin, ResultMin) = False Then
        FEMTET.ShowLastError
    End If

    '------- 任意座標の計算結果の取得 -------
    Dim Value() As New CComplex

    If Gogh.Galileo.GetVectorAtPoint(0, 0, 0, Value()) = False Then
        FEMTET.ShowLastError
    End If

    ' 複数の座標の結果をまとめて取得する場合は、MultiGetVectorAtPoint関数をご利用ください。

End Sub
```

10に変更

サンプルコードでは結果取得を実施していますが、その結果をどこにも出力していません。

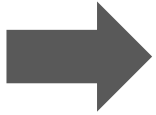
以下のコードを末尾に追記して結果をワークシートに出力してみましょう

```
Cells(1, 1) = ResultMax
Cells(2, 1) = PosMax(0)
Cells(3, 1) = PosMax(1)
Cells(4, 1) = PosMax(2)

Cells(6, 1) = ResultMin
Cells(7, 1) = PosMin(0)
Cells(8, 1) = PosMin(1)
Cells(9, 1) = PosMin(2)

Cells(11, 1) = Value(0).Real
Cells(12, 1) = Value(1).Real
Cells(13, 1) = Value(2).Real
```

4-6 結果取得までマクロ実行する

Murata Software

```
'------- メッシュの生成 -------
'<<<<<<< メッシュを生成する場合は以下
'Gaudi.Mesh

'------- 解析の実行 -------
'<<<<<<< 解析を実行する場合は以下のコ
'FEMTET.Solve

'------- 解析結果の抽出 -------
'<<<<<<< 計算結果を抽出する場合は以下
'SamplingResult
```



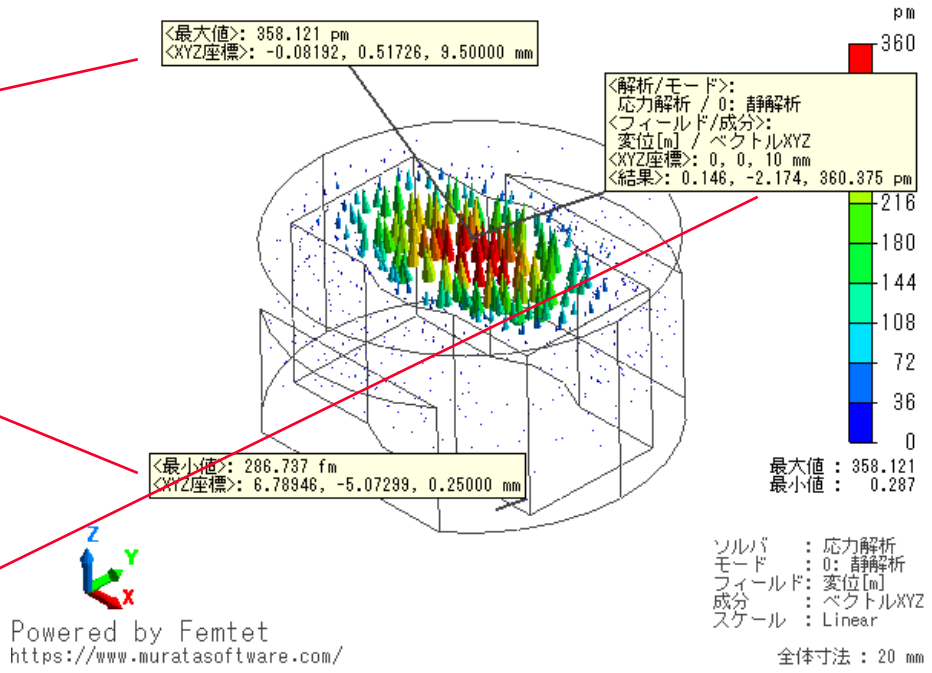
```
'------- メッシュの生成 -------
'<<<<<<< メッシュを生成する場合は以下のコメン
Gaudi.Mesh

'------- 解析の実行 -------
'<<<<<<< 解析を実行する場合は以下のコメントを
FEMTET.Solve

'------- 解析結果の抽出 -------
'<<<<<<< 計算結果を抽出する場合は以下のコメン
SamplingResult
```

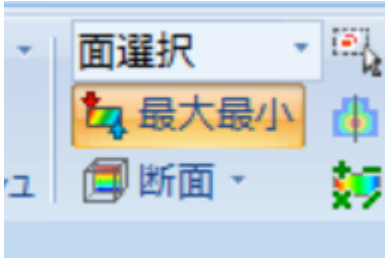
FemtetMain関数の終盤の
メッシュの生成、解析の実行、計算結果の抽出の処理について
上の３行のコメントを外します。
「'」 を消します。

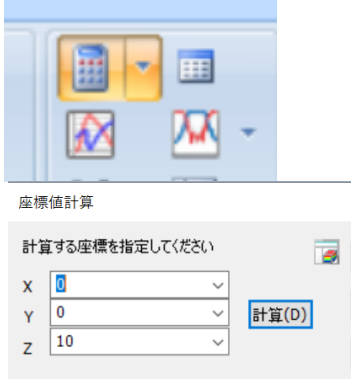
4-6 結果取得までマクロ実行する

Murata Software

	A
1	3.58E-10
2	-0.08192
3	0.517257
4	9.5
5	
6	2.87E-13
7	6.789456
8	-5.07299
9	0.25
10	
11	1.46E-13
12	-2.2E-12
13	3.6E-10
14	







結果表示の結果とマクロ取得の結果が一致している

【VBA文法】変数の型について

Murata Software

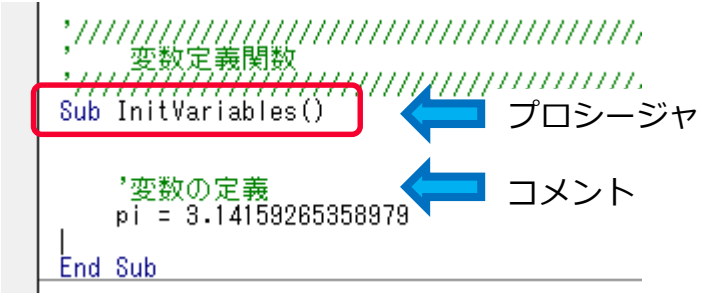
VBAでプログラムを作成する際に、数値のデータや文字列のデータ等を保存する入れ物として変数を使用します。
変数がどのようなデータを扱うかは指定する「型」によって変わります。
以下の表はVBAでコードを書く上でよく使用する型をまとめたものです。

データ型		使用メモリ	値
ブール型	Boolean	2バイト	True または False
整数型	Integer	2バイト	-32,768 ～ 32,767 の整数
長整数型	Long	4バイト	-2,147,483,648～2,147,483,647の整数
倍精度浮動小数点型	Double	8バイト	（負）-1.79769313486231E308～ -4.94065645841247E-324 （正）4.94065645841247E-324～ 1.79769313486231E308
文字列型	String	10バイト+文字列の長さ	文字データ
オブジェクト型	Object	4バイト	オブジェクトを参照するデータ
バリアント型	Variant		全ての値

【VBA文法】プロシージャとコメント

Murata Software

Sub FemtetMain()、Sub InitVariables()といった処理をまとめたものをプロシージャと呼びます。Sub ～()という定義の場合、プロシージャ内の処理を実行するだけ、Function ～()という定義の場合、プロシージャ内の処理を実行し呼び出し元に戻り値(例えばプロシージャ内で足し算を行いその結果を戻り値とする)を返します。



「'」を入れると以降の文字列はコメント文扱いとなり処理されません。プロシージャ内の処理内容などを記述しておくと良いです。

【VBA文法】セルとの数値の出入力

```
Cells(1, 1) = ResultMax
Cells(2, 1) = PosMax(0)
Cells(3, 1) = PosMax(1)
Cells(4, 1) = PosMax(2)

Cells(6, 1) = ResultMin
Cells(7, 1) = PosMin(0)
Cells(8, 1) = PosMin(1)
Cells(9, 1) = PosMin(2)

Cells(11, 1) = Value(0).Real
Cells(12, 1) = Value(1).Real
Cells(13, 1) = Value(2).Real
```

行番号、列番号

	A
1	3.58E-10
2	-0.08192
3	0.517257
4	9.5
5	
6	2.87E-13
7	6.789456
8	-5.07299
9	0.25
10	
11	1.46E-13
12	-2.2E-12
13	3.6E-10
14	

Cells(行番号、列番号)
で各セルに値を出力したり、

逆に変数にセルの値や文字を入力することもできる

例：
A = Cells(1,1)

【VBA文法】If文

Murata Software

If文は条件によって処理を分けたい場合に使用します。
条件分岐で処理を柔軟に制御できるようになります。

```
'------- 新規プロジェクト -------
If FEMTET.OpenNewProject() = False Then
    FEMTET.ShowLastError
End If
```

【書式】
基本的な書式は“If 条件 Then 処理 End If”となります。
条件を満たすとき処理を実行するという事になります。
←のコードの場合OpenNewProject()の戻り値がFalseの時、
Femtet.ShowLastErrorを実行するという処理になります。

【2分岐】

```
If a = 1 Then
    b = 1
Else
    b = 2
End If
```

【3分岐】

```
If a = 1 Then
    b = 1
ElseIf a = 2 Then
    b = 2
Else
    b = 3
End If
```

【複雑な条件分岐】
2分岐の場合⇒“If 条件 Then 処理1 Else 処理2 End If”
条件を満たすとき処理1、満たさないとき処理2を実行します。

3分岐以上の場合⇒“If 条件1 Then 処理1 ElseIf 条件2 Then 処理2 Else ~ End If”
条件1を満たすとき処理1、条件2を満たすとき処理2～という流れで処理を実行します。

【VBA文法】If文

Murata Software

【複数条件を満たす】

```
If a = 1 And b = 1 Then
    X = 1
End If
```

【複雑な条件分岐】
複数の条件を満たす場合⇒“If 条件1 And 条件2 Then 処理 End If”
条件1と条件2を両方満たす場合、処理を実行します。

【いずれかの条件を満たす】

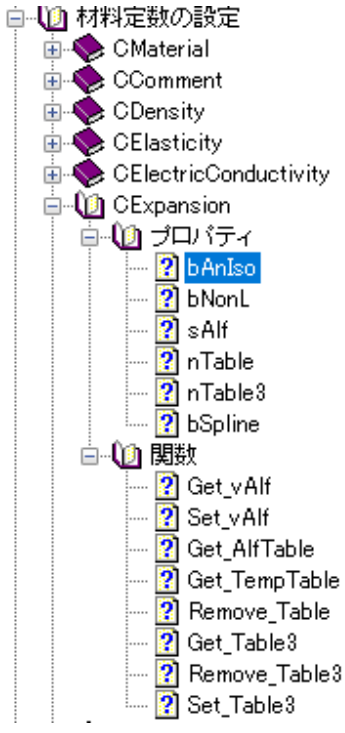
```
If a = 1 Or b = 1 Then
    X = 1
End If
```

いずれかの条件を満たす場合⇒“If 条件1 Or 条件2 Then 処理 End If”
条件1あるいは条件2を満たすとき処理を実行します。

【VBA文法】関数とプロパティ

Murata Software

関数とプロパティではコードの記述が異なります。線膨張係数の例を示します。



sAlfプロパティ

定義

sAlf As Double
値の取得 ○
値の設定 ○

引数

引数はありません

解説

線膨張係数を設定、取得します。

実行例

```
Dim Femtet As New CFemtet
Dim Mat As CMaterial

If Femtet.OpenNewProject() = False Then
    Femtet.ShowLastError
End If

'Femtet.Materialを変数Matに代入して使いやすくする
Set Mat = Femtet.Material

Mat.Add "MaterialA"

Mat.Expansion(0).sAlf = 0.000001
```

単純に代入または参照

Get_vAlf関数

マクロヘルプより

定義

Get_vAlf (Index As Long) As Double

戻り値

異方性線膨張係数（温度依存性なし）の指定Index値

引数

Index

異方性線膨張係数（温度依存性なし）のインデックス
インデックス値と各成分は以下のように対応しています。

インデックス値	成分
0	x
1	y
2	z

解説

異方性線膨張係数（温度依存性なし）の指定Index値を取得します。

実行例

```
Dim Femtet As New CFemtet
Dim Mat As CMaterial
Dim dData As Double

If Femtet.LoadProject("C:¥Test¥test.femprj", True) = False Then
    Femtet.ShowLastError
End If

'Femtet.Materialを変数Matに代入して使いやすくする
Set Mat = Femtet.Material

dData = Mat.Expansion(0).Get_vAlf(0)
```

引数を指定して
取得または設定

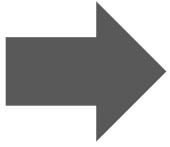
５章の狙い

５章ではマクロコードを変更して応力解析から熱伝導解析の条件に変更していきます。

少しずつ変更を加えて、その変更が実際に反映されているかを確認することで、少しずつマクロを理解を深めていくことができます。

5章に入る前の準備事項

```
    '------- メッシュの生成 -------
    '<<<<<<< メッシュを生成する場合は以下のニ
→   Gaudi.Mesh

    '------- 解析の実行 -------
    '<<<<<<< 解析を実行する場合は以下のコメン
→   FEMTET.Solve

    '------- 解析結果の抽出 -------
    '<<<<<<< 計算結果を抽出する場合は以下のニ
→   SamplingResult
```



```
'------- メッシュの生成 -------
'<<<<<<< メッシュを生成する場合は以下
'Gaudi.Mesh

'------- 解析の実行 -------
'<<<<<<< 解析を実行する場合は以下のニ
'FEMTET.Solve

'------- 解析結果の抽出 -------
'<<<<<<< 計算結果を抽出する場合は以下
'SamplingResult
```

FemtetMain関数の終盤の
メッシュの生成、解析の実行、計算結果の抽出の処理について
上の３行をコメントアウトします。「'」を最初に付けます。

5.　データベース設定を変更する

実際にコードを少しずつ変更し動作確認をして
理解を深めていきましょう

5-1　解析条件の変更（ソルバー選択）
5-2　材料定数の編集（熱伝導率の編集）
5-3　境界条件の変更（温度境界の設定）
5-4　境界条件の変更（外部境界の設定）

5-1　解析条件の変更（ソルバー選択）

Murata Software

Als.AnalysisTypeで解析条件のソルバを指定することができます。
「=」まで消した後、あらためて「=」を入力するとENUM値一覧が表示されます。

```
Sub AnalysisSetUp()

    '------- 変数にオブジェクトの設定 -------
    Set Als = FEMTET.Analysis

    '------- 解析条件共通(Common) -------
    Als.AnalysisType |

    '------- 磁場(Gauss) -------
    Als.Gauss.b2ndEdgeElement = True

    '------- 電磁波(Hertz) -------
```

```
Sub AnalysisSetUp()

    '------- 変数にオブジェクトの設定 -------
    Set Als = FEMTET.Analysis

    '------- 解析条件共通(Common) -------
    Als.AnalysisType =

    '------- 磁場(G
    Als.Gauss.b2ndE

    '------- 電磁波
    Als.Hertz.b2ndE

    '------- 調和解
```

BERNOULLI_C
COULOMB_C
GAL_LIFE_C
GAL_STATIC_C
GALILEO_C
GAUSS_C
HERTZ_C

ENUM値「WATT_C」（熱伝導解析）
に指定してみます。

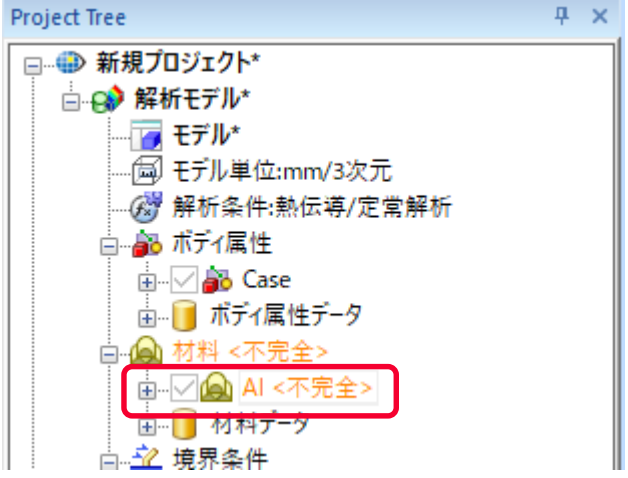
5-1　解析条件の変更（動作確認）

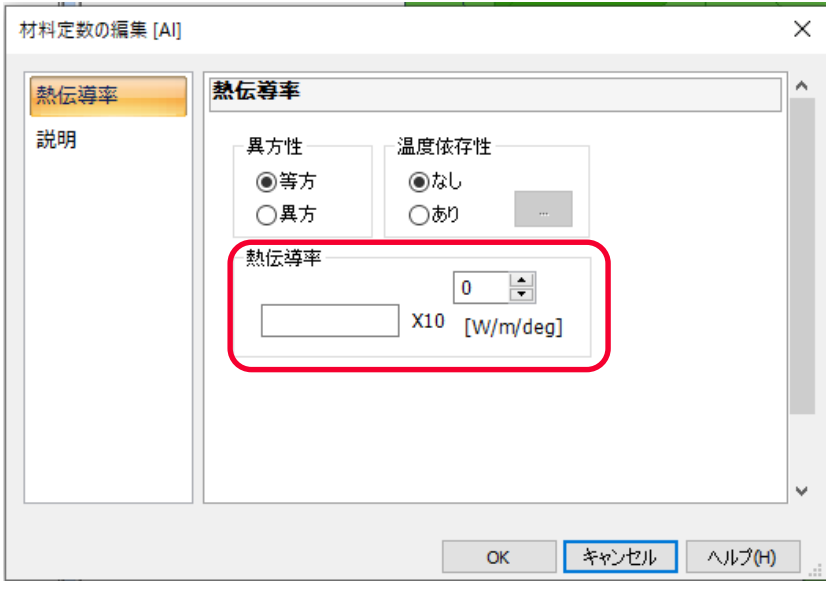
Murata Software

Als.AnalysisTypeのENUM値を変更したら、マクロを実行します。
実行後、Femtet画面で解析条件を確認すると、熱伝導解析に替わっているはずです。

5-2　材料定数の編集（熱伝導率の編集）

次に、材料定数を変更してみます。
解析条件を熱伝導解析に変更したので、熱伝導率を設定する必要があります。





Femtet画面で確認すると、材料「Al」が不完全となり、熱伝導率が空になっています。

5-2　材料定数の編集（熱伝導率の編集）

Murata Software

材料「AI」の内容を設定しているMaterialSetUp_AIプロシージャに熱伝導率を設定するコードを追加します。
熱伝導率はCThermalConductivityクラスのsRmdプロパティで指定します。

```
Sub MaterialSetUp_AI()
    '------- MaterialのIndexを保存する変数 -------
    Dim Index As Integer

    '------- Materialの追加 -------
    Mtl.Add "AI"

    '------- Material Indexの設定 -------
    Index = Mtl.Ask("AI")

    Mtl.ThermalConductivity(Index).sRmd = 1.5

    '------- 線膨張係数(Expansion) -------
    Mtl.Expansion(Index).sAlf = (0#)
    Mtl.Expansion(Index).Set_vAlf 0, (0#)
    Mtl.Expansion(Index).Set_vAlf 1, (0#)
    Mtl.Expansion(Index).Set_vAlf 2, (0#)
```

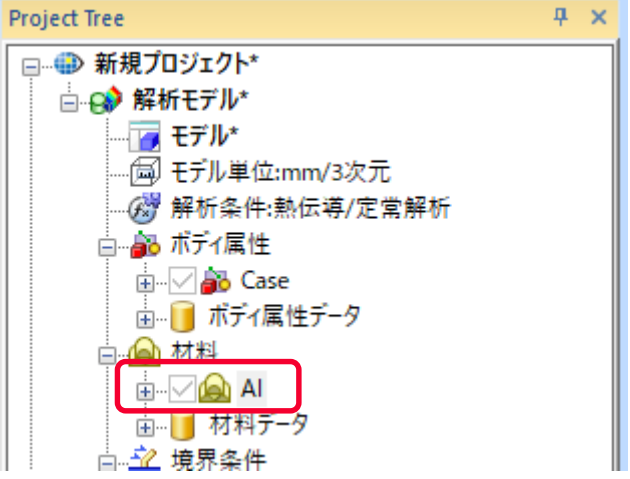
CThermalConductivityプロパティリスト

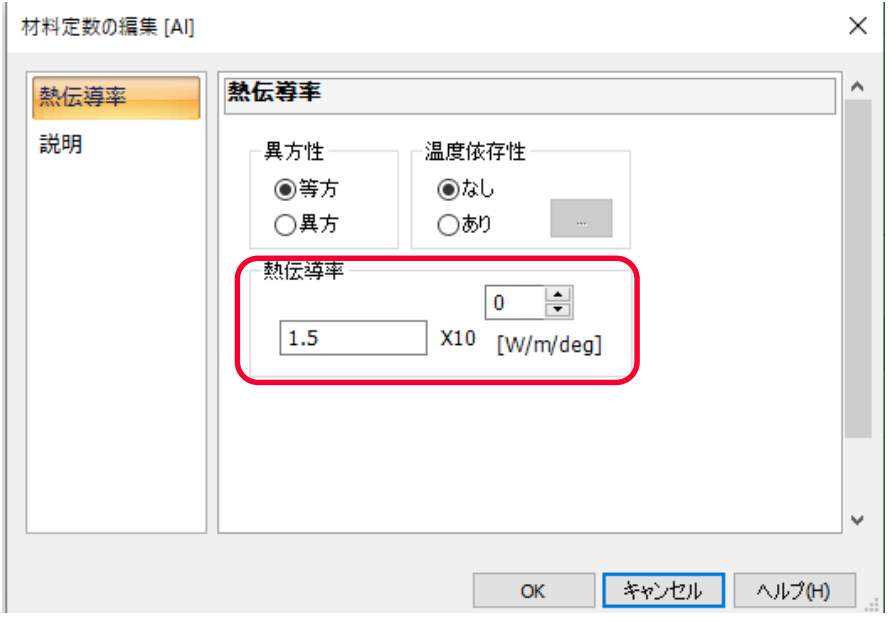
プロパティ名	説明
bAnIso	等方/異方
bNonL	温度依存性 あり/なし
nTable	等方性温度依存性テーブルのデータ数
nTable6	異方性温度依存性テーブルのデータ数
bSpline	温度依存性テーブルの近似方法
sRmd	熱伝導率

Indexの値を取得した以降にこのコードを追加します。

5-2　材料定数の編集（動作確認）

sRmdプロパティを追加したらマクロを実行します。
材料AIに熱伝導率が設定されている事が確認できます。





5-3　境界条件の変更（温度境界の設定）

次に境界条件を変えてみます。熱伝導解析に変更しているので、境界条件Fix、Pressureを温度の条件に変えてみましょう。境界条件の種類はConditionプロパティ、温度はCWallTempクラスのTempプロパティで指定します。

```
Sub BoundarySetUp_Fix()
    '------- BoundaryのIndexを保存する変数 -------
    Dim Index As Integer

    '------- Boundaryの追加 -------
    Bnd.Add "Fix"

    '------- Boundary Indexの設定 -------
    Index = Bnd.Ask("Fix")

    Bnd.Thermal(Index).Condition = TEMPERATURE_C
    Bnd.Thermal(Index).WallTemp.Temp = 25

Sub BoundarySetUp_Pressure()
    '------- BoundaryのIndexを保存する変数 -------
    Dim Index As Integer

    '------- Boundaryの追加 -------
    Bnd.Add "Pressure"

    '------- Boundary Indexの設定 -------
    Index = Bnd.Ask("Pressure")

    Bnd.Thermal(Index).Condition = TEMPERATURE_C
    Bnd.Thermal(Index).WallTemp.Temp = 50
```

CWallTempプロパティリスト

プロパティ名	説明
Temp	温度

Conditionプロパティに「TEMPERTURE_C」
Fixの温度を25度、Pressureの温度を50度に設定します。

5-3　境界条件の変更（外部境界の設定）

Murata Software

さらに外部境界条件を放熱・環境輻射、自然対流（係数自動計算）の条件に変えてみましょう。自然対流（係数自動計算）の指定はbConAutoプロパティで指定します。

```
Sub BoundarySetUp_RESERVED_default()
    '------- BoundaryのIndexを保存する変数 -------
    Dim Index As Integer

    '------- Boundaryの追加 -------
    Bnd.Add "RESERVED_default"

    '------- Boundary Indexの設定 -------
    Index = Bnd.Ask("RESERVED_default")

    Bnd.Thermal(Index).Condition = HEAT_TRANSFER_C
    Bnd.Thermal(Index).bConAuto = True

    '------- 室温_環境温度(RoomTemp) -------
    Bnd.Thermal(Index).RoomTemp.Temp = 25
```

Conditionプロパティに「HEAT_TRANSFER_C」、
bConAutoプロパティに「True」を指定します。
室温(RoomTemp.Tempプロパティ)を25度に変えておきます。

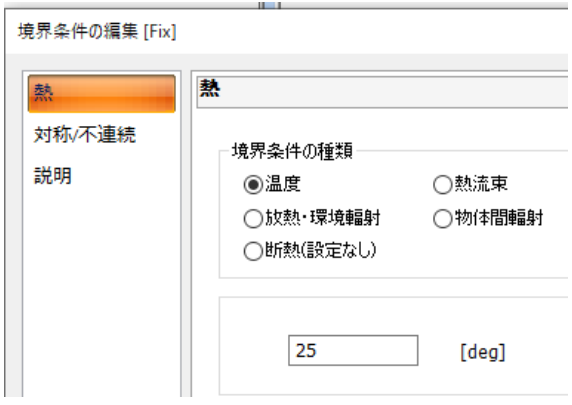
CThermalプロパティリスト

プロパティ名	説明
Condition	境界条件の種類
WallTemp	温度（CWallTemp）オブジェクト
HeatFlux	熱流束の設定
ThermRType	抵抗の入力形式
ThermR	熱抵抗、面積当たり熱抵抗
ThermRCond	熱抵抗の熱伝導率成分
ThermRThickness	熱抵抗の厚み成分
bOpen	熱抵抗設定面を、応力解析で開放にするかどうか
bInsulate	熱抵抗設定面を、電場解析で絶縁にするかどうか
bH	熱伝達係数の考慮の有無
bFcon	強制対流の考慮の有無
bConAuto	自然対流（係数自動計算）の考慮の有無
bCon	自然対流の考慮の有無

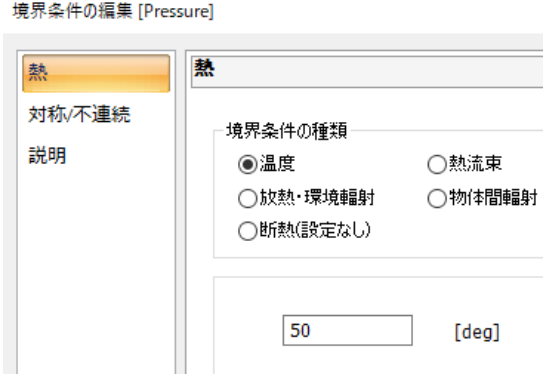
5-3　境界条件の変更（動作確認）

境界条件の内容を変更したらFemtetMainプロシージャを実行します。
境界条件Fix、Pressureが温度の条件、外部境界条件が放熱・環境輻射の条件に替わっています。

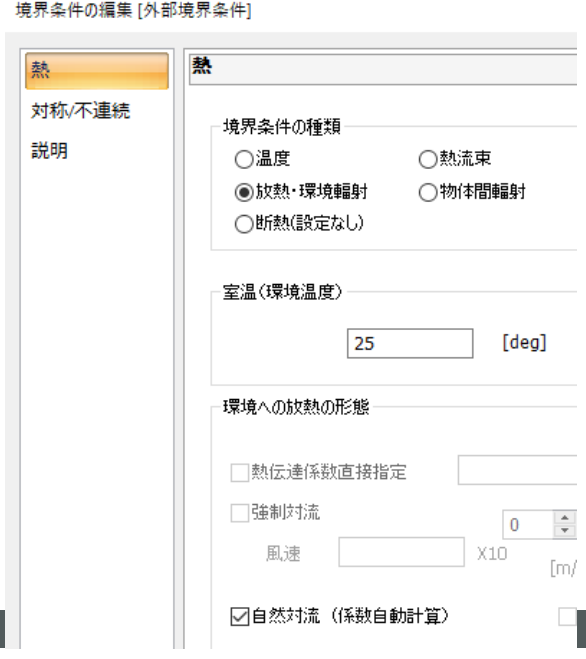
「Fix」



「Pressure」



「外部境界条件」



６章の狙い

６章ではマクロコードを変更してモデル寸法を変更します。

MakeModel関数のどの部分が変更したい寸法に対応するのかを
突き止めてからコードを変更します。

6.モデルの寸法を変更する

Murata Software

6-1.変えたい寸法の提示
6-2.モデルの履歴からの確認
6-3.コードを変更

【VBA文法】配列

Murata Software

配列は変数をまとめておく入れ物のようなものです。例えば、コピーコマンドの実行結果として複数のボディが生成される場合に、結果のボディの格納先として配列が使用されます。

```
Sub TestFunc()

    Dim X(6) As String

    X(0) = "sunday"
    X(1) = "monday"

End Sub
```

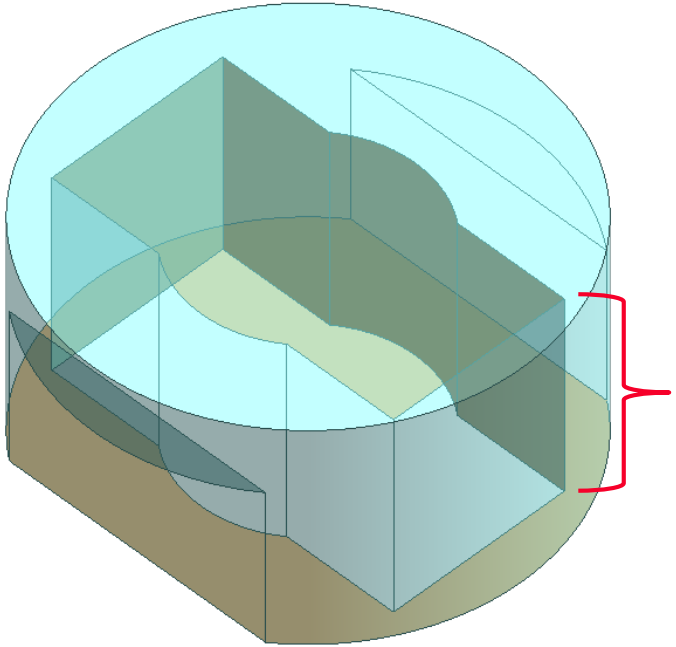
【宣言方法と値の設定】
配列の宣言方法は←のコードのように、“配列名(要素数) As 変数の型”という書式で宣言します。
要素数は０から数えるので、左のコードだとString型のデータを7つ(0～6)もつ配列Xを定義することになります。
配列に値を設定するには”配列名(何番目) = 値”という書式で設定します。

```
Sub TestFunc()

    Dim X(6) As String
    Dim Size As Long

    Size = 100
    ReDim X(Size)

End Sub
```

【要素数の変更】
配列の要素数を後で変更する場合、“Redim 配列名(要素数)”という書式で設定します。
Redimで要素数を変更すると要素内のデータはクリアされます。
データを残しつつ要素数を変更したい場合は、 “Redim Preserve 配列名(要素数)”という書式で設定してください。

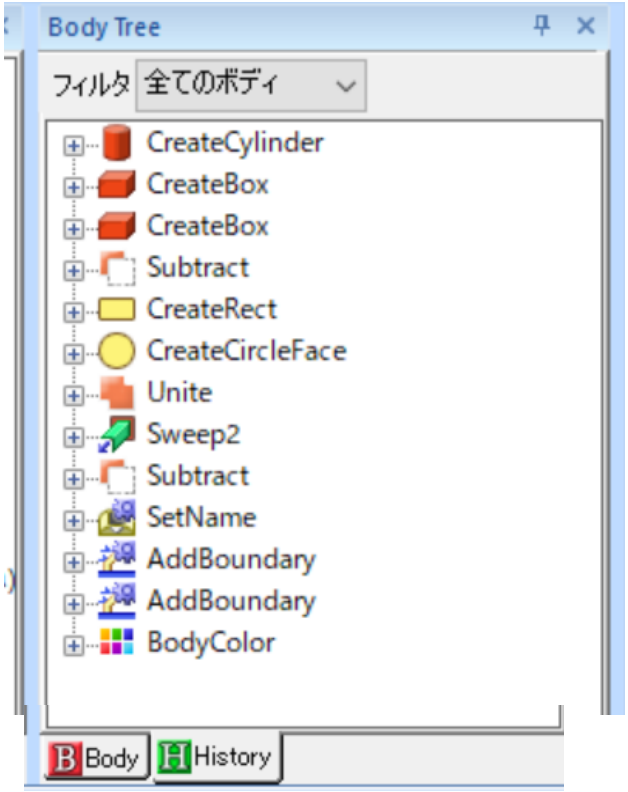
6-1.変えたい寸法の提示

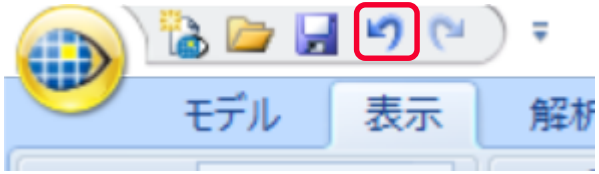
作成されるモデルの穴部分の深さをコードを書き換えて変更してみましょう。



この部分の深さを変更します。

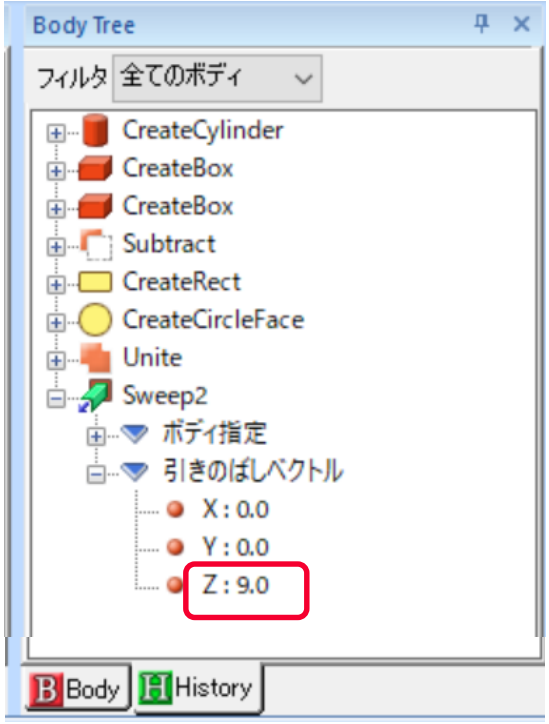
6-2.モデルの履歴からの確認

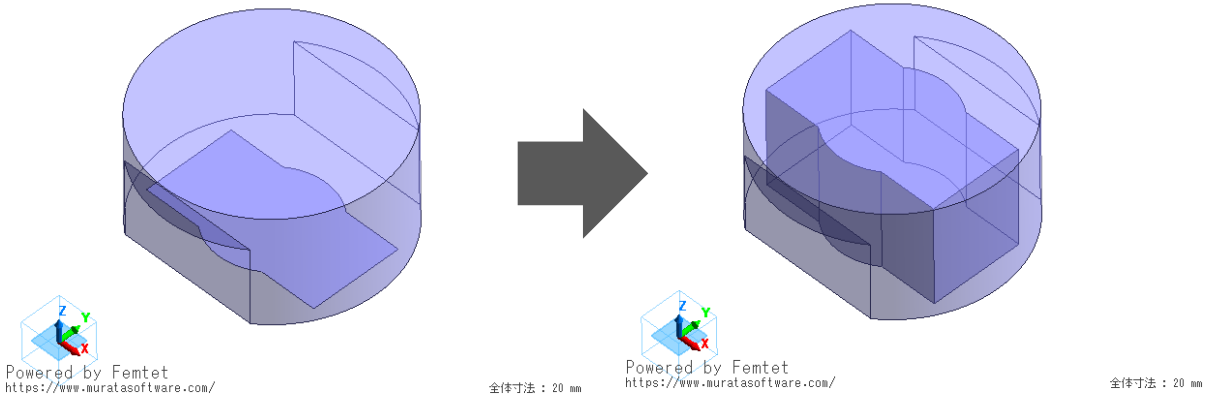




Undoして穴の深さに影響する作図処理を特定します。

6-2.モデルの履歴からの確認





Uniteの後のSweep2が穴の深さに影響していることが分かります。

実際に引き伸ばしベクトルのZ成分を変更して確認します。

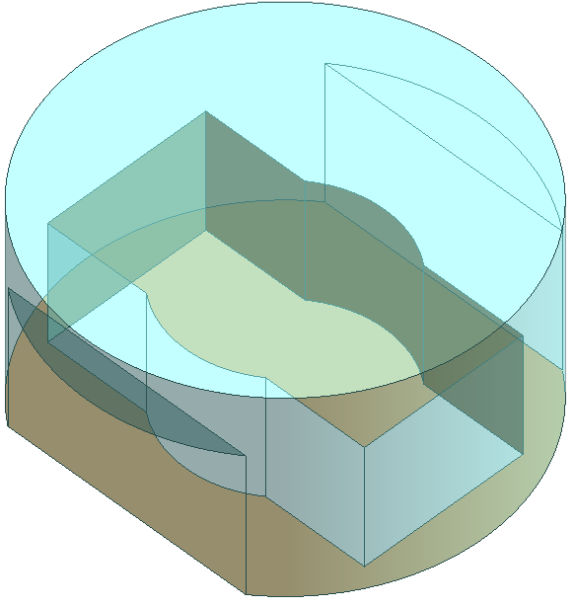
6-3.コードを変更

穴部分の深さに該当するのは、Sweep2関数のVector0の３つ目の値になります。

```
'------- Sweep2 -------
Dim Vector0 As New CGaudiVector
Vector0.SetCoord 0#, 0#, 9#
Body(6).Sweep2 Vector0
```

Vector0の値を0,0,6に変更します。

```
'------- Sweep2 -------
Dim Vector0 As New CGaudiVector
Vector0.SetCoord 0#, 0#, 6
Body(6).Sweep2 Vector0
```



Vector0の値を変えたらマクロを実行します。
穴部分の深さが浅くなっています。

7.メッシュサイズを変更し解析実行する

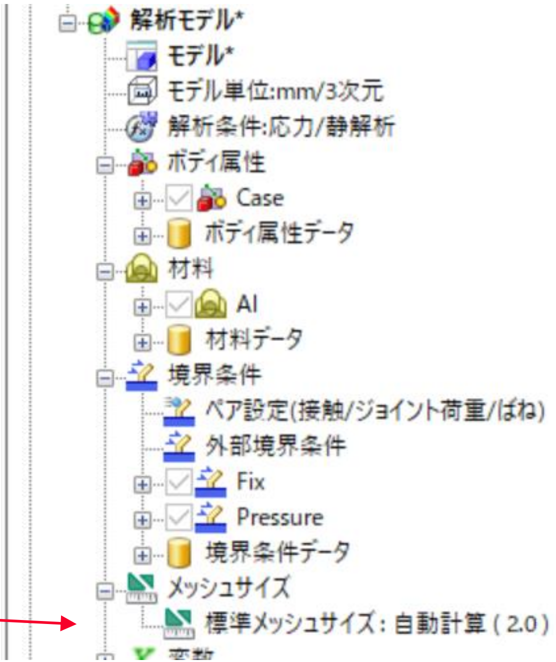
7-1 メッシュサイズを変更する
7-2 解析実行する

7-1.メッシュサイズを変更する

Murata Software

FemtetMain関数の標準メッシュサイズ設定を変更します。

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Sub FemtetMain()
    '------- Femtet自動起動 (不要な場合やExcelで実行しない場合は下行をコメ
    Workbooks("FemtetRef.xla").AutoExecuteFemtet

    '------- 新規プロジェクト -------
    If FEMTET.OpenNewProject() = False Then
        FEMTET.ShowLastError
    End If

    '------- 変数の定義 -------
    InitVariables

    '------- データベースの設定 -------
    AnalysisSetUp
    BodyAttributeSetUp
    MaterialSetUp
    BoundarySetUp

    '------- モデルの作成 -------
    Set Gaudi = FEMTET.Gaudi
    MakeModel

    '------- 標準メッシュサイズの設定 -------
    '<<<<<<< 自動計算に設定する場合は-1を設定してください >>>>>>>
    Gaudi.MeshSize = 2#
```

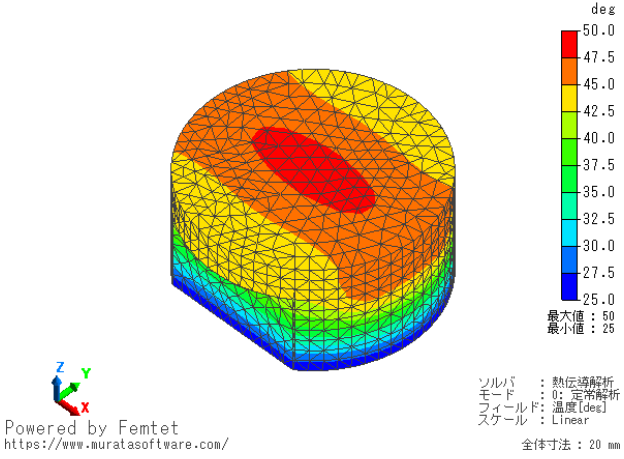


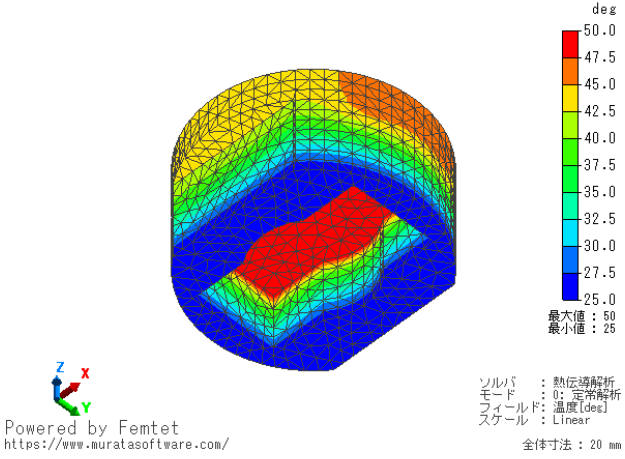
例えば２→1へ変更します。

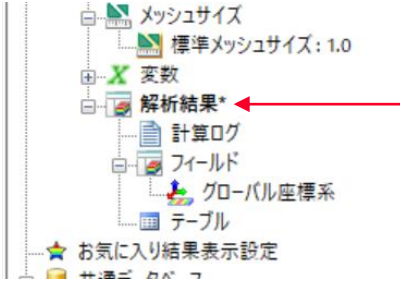
7-2.メッシュ生成し解析実行する

FemtetMain関数のメッシュ生成、解析実行のコメントを外しマクロを起動します。

```
'------- メッシュの生成 ------
'<<<<<<< メッシュを生成する場合
Gaudi.Mesh

'------- 解析の実行 -------
'<<<<<<< 解析を実行する場合は
FEMTET.Solve

'------- 解析結果の抽出 ------
'<<<<<<< 計算結果を抽出する場合
'SamplingResult
```







ダブルクリックで結果表示

8.解析結果の抽出内容を変更する

8-1. 取得するフィールドを温度に変更する
8-2. 温度の最大値、最小値を取得するコードに変更する
8-3. 指定座標の温度を取得するコードに変更する
8-4. 結果抽出までマクロ実行する

【補足】ポテンシャル、ベクトル、テンソル

Femtetのフィールド分布の結果はその物理量によってポテンシャル、ベクトル、テンソルに分かれてきます。取得したいフィールドに応じた適切なフィールド設定が必要です。

	熱伝導解析（Watt)	応力解析（Galileo)
ポテンシャル※	温度	相当応力
ベクトル	熱流束	変位
テンソル	なし	応力、ひずみ

※スカラー量と同じです

マクロヘルプより

8-1. 取得するフィールドを温度に変更する

SamplingResult関数を編集します。
取り出す計算結果を温度に変更しましょう。
取り出す結果の内容が応力解析の変位になっているので、
熱伝導解析の温度に変更します。

```
'------- フィールドの設定 -------
Gogh.Galileo.Vector = GALILEO_DISPLACEMENT_C
```

```
'------- フィールドの設定 -------
Gogh.Watt.Potential = WATT_TEMPERATURE_C
```

熱伝導解析の温度を取得するには、Watt.Potentialプロパティを
「WATT_TEMPERTURE_C」に指定します。

Potentialプロパティ

定義

Potential As WATT_POT_T
値の取得 ○
値の設定 ○

引数

ありません

解説

ポテンシャルの種類を設定、取得します。
WATT_POT_T で定義された種類を指定してください。

WATT_TEMPERATURE_C	温度[deg]
WATT_RADIOSITY_C	射度[W/m2]
WATT_FLUID_HEAT_TRANSFER_C	対流の熱伝達係数
[W/m2/deg]	
WATT_HEAT_DENSITY_C	発熱密度[W/m2]
WATT_TURBULENT_THERMAL_CONDUCTIVITY_C	乱流熱伝導率[W/m/deg]
WATT_RESIDUAL_C	残差[W]

8-2. 温度の最大値、最小値を取得するコードに変更する

次に温度の最大値、最小値の値を取得します。

```
'------- 最大値の取得 -------
Dim PosMax() As Double '最大値の座標
Dim ResultMax As Double ' 最大値

If Gogh.Galileo.GetMAXVectorPoint(VEC_C, CMPX_REAL_C, PosMax, ResultMax) = False Then
    FEMTET.ShowLastError
End If

'------- 最小値の取得 -------
Dim PosMin() As Double '最小値の座標
Dim ResultMin As Double '最小値

If Gogh.Galileo.GetMINVectorPoint(VEC_C, CMPX_REAL_C, PosMin, ResultMin) = False Then
    FEMTET.ShowLastError
End If
```

```
'------- 最大値の取得 -------
Dim PosMax() As Double '最大値の座標
Dim ResultMax As Double ' 最大値

If Gogh.Watt.GetMAXPotentialPoint(CMPX_REAL_C, PosMax, ResultMax) = False Then
    FEMTET.ShowLastError
End If

'------- 最小値の取得 -------
Dim PosMin() As Double '最小値の座標
Dim ResultMin As Double '最小値

If Gogh.Watt.GetMINPotentialPoint(CMPX_REAL_C, PosMin, ResultMin) = False Then
    FEMTET.ShowLastError
End If
```

温度の最大値、最小値は、Watt.GetMAXPotentialPoint()、Watt.GetMINPotentialPoint()で取得できます。

8-3. 指定座標の温度を取得するコードに変更する

Murata Software

次に任意の座標の温度を取得します。

```
'------- 任意座標の計算結果の取得 -------
Dim Value() As New CComplex

If Gogh.Galileo.GetVectorAtPoint(0, 0, 0, Value()) = False Then
    FEMTET.ShowLastError
End If
```

```
'------- 任意座標の計算結果の取得 -------
Dim Value As New CComplex

If Gogh.Watt.GetPotentialAtPoint(0, 0, 10, Value) = False Then
    FEMTET.ShowLastError
End If
```

```
Cells(11, 1) = Value(0).Real
Cells(12, 1) = Value(1).Real
Cells(13, 1) = Value(2).Real
```

```
Cells(11, 1) = Value.Real
```

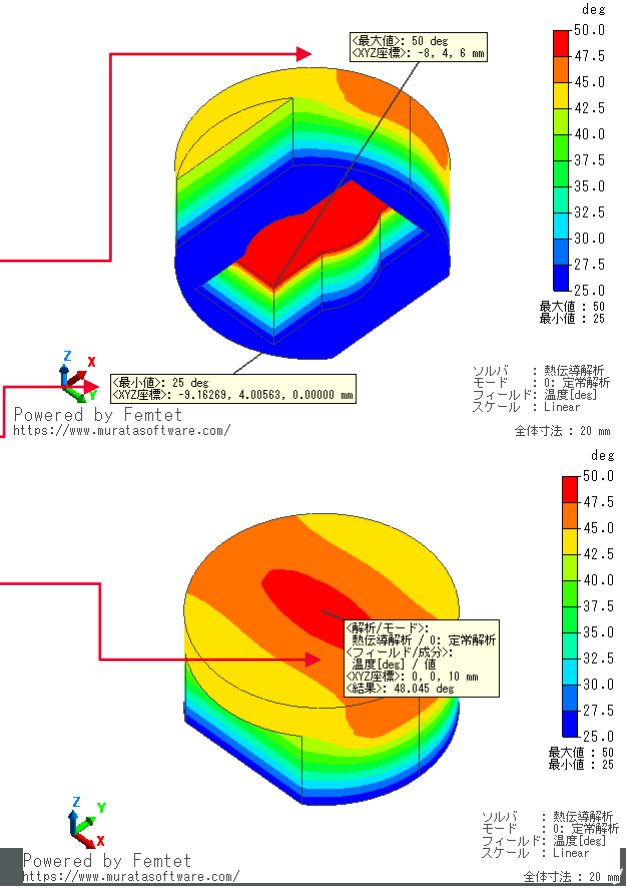
任意の座標の温度の結果は、Watt.GetPotentialAtPoint() で取得できます。
戻り値は配列から変数に変更、取り出す座標を0,0,10に変更します。
結果をセルに表示する処理も上記のように変更します。

8-4. 結果抽出までマクロ実行する

```
'------- 解析結果の抽出 ------
'<<<<<<< 計算結果を抽出する場
'SamplingResult
```

```
'------- 解析結果の抽出
'<<<<<<< 計算結果を抽出
SamplingResult
```

	A
	50
2	-8
3	4
4	6
5	
6	25
7	-9.1627
8	4.00563
9	0
10	
11	48.0454
12	



マクロ編集のコツ

Murata Software

編集対象	編集のコツ
データベース設定	GUIでの設定内容とコードを見比べることで 編集ポイントを特定する
モデル作成	履歴（ヒストリー）とコードを見比べることで 編集ポイントを特定する
結果取得	サンプルコードを編集する マクロヘルプのサンプルコードを参考にする 不明点はユーザーサポートに問い合わせる
一般的なVBAの 記述方法	Web検索すると様々な記述例を知ることができ るのでそれらを参考にする。

付録

VBAの文法とサンプルマクロのご紹介

【VBA文法】For文

Murata Software

For文は同じ処理を繰り返したい場合に使用します。

【For文】

```
Dim i As Integer
For i = 0 To 2
    MsgBox "iの値は" & i
Next i
```

【For文】iを 2 づつ増やす

```
Dim i As Integer
For i = 0 To 4 Step 2
    MsgBox "iの値は" & i
Next i
```

【For文の書式】
基本的な書式は“For 変数 = A To B 処理 Next 変数”となります。
変数がAからBになるまで処理を実行し続けます。
例えば←のコードだと、iの値が2になるまでiの値をメッセージ表示します。
この書式の場合、変数の値は1づつ増えていきます。
変数の値の増やし方を変えたい場合、
“For 変数 = A To B Step C 処理 Next 変数”と記述します。
この書式の場合、変数の値はCづつ増えていきます。

【VBA文法】Do～While文

Murata Software

Do~While文は、条件を満たす間、同じ処理を繰り返したい場合に使用します。

【Do~While文】

```
Dim i As Integer
i = 0
Do While i < 5
    MsgBox "iの値は" & i
    i = i + 1
Loop
```

【Do~While文の書式】
基本的な書式は“Do While 条件 処理 Loop”となります。
条件を満たす間、処理を実行し続けます。
例えば←のコードだと、iの値が5未満の間、iの値をメッセージ表示します。

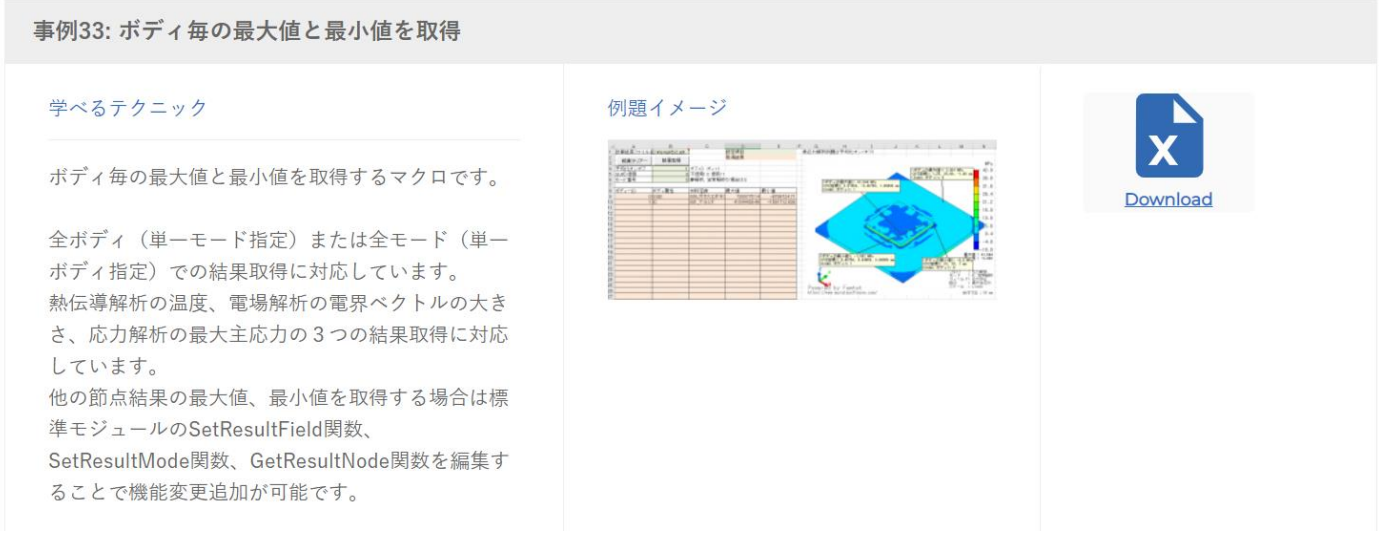
※For文やDo~While文などの繰り返し処理を実行するコードは、
書き方を間違えて常に条件を満たすような処理にしてしまうと、
無限ループを起こしてしまうので気を付けましょう。

サンプルマクロについて

Murata Software

ムラタソフトウェアのウェブサイト上にExcelで動作可能なサンプルマクロを多数公開しています。サンプルマクロのコードを編集してカスタマイズする事も可能なので是非ご参照ください。

https://www.muratasoftware.com/support/macro/



ーおわりー