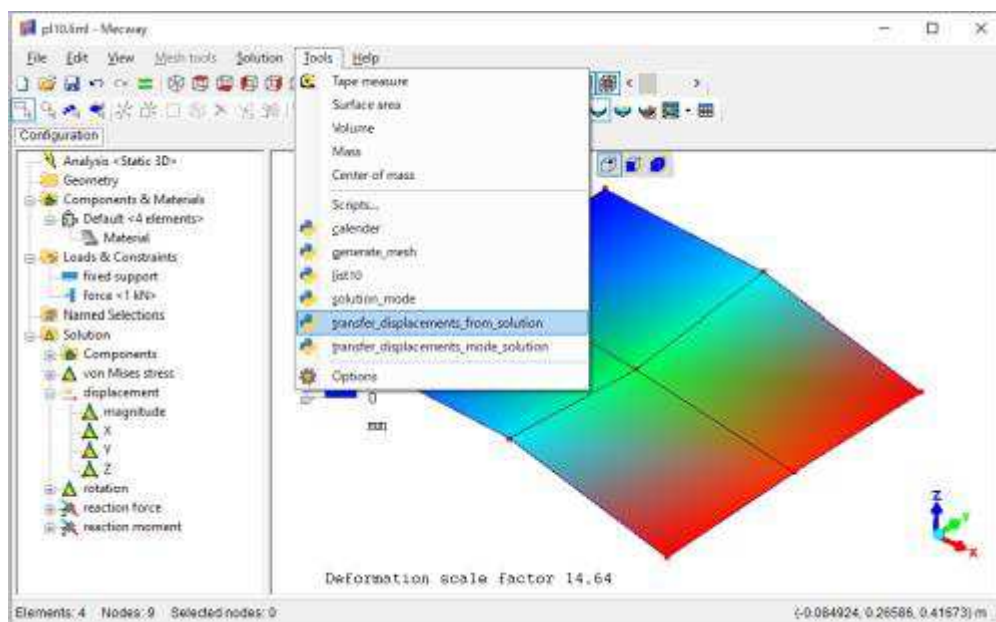


# Python スクリプト・インタープリタ for Mecway FEA



Mecway Finite Element Analysis

Version 23.0

# Python スクリプト・インタープリタ

ソフトウェアに組み込まれた Mecway スクリプト・インターフェイスは、Python (IronPython2.7) ライブラリが使用されます。この解説書では、Mecway スクリプトの API (アプリケーション・プログラミング・インターフェイス) のリファレンス及び使用例を示します。

## File

### **mw.file\_name()**

Returns                str                                開いている liml の絶対パスとファイル名

### **mw.import\_file(path)**

既に関いていることへその内容を合併して、ファイルを輸入します。解決データおよびテーブルが除外されます。Mecway が輸入することができるスクリプト(.py 拡張)以外のファイル・タイプがすべて支援されます。

path                                str

### **mw.open(path)**

ファイルを開きます。Mecway が開くことができるスクリプト(.py 拡張)以外のファイル・タイプがすべて支援されます。

path                                str

### **mw.save()**

現在、開いているファイルにモデルを保存します。

### **mw.save\_as(path)**

指定ファイルのモデルを保存します。

path                                str

## Edit

### **mw.selected\_elements()**

Returns                List[int]                                現在選択されている要素の要素番号をリストします。

### **mw.selected\_faces()**

Returns                List[FaceId]                現在選択されている Face Id をリストします。

### **mw.selected\_nodes()**

Returns                List[int]                現在選択されているノードのノード番号をリストします。

### **solution.selected\_elements()**

Returns                List[int]                現在選択されている要素の要素番号です。

### **solution.selected\_faces()**

Returns                List[FaceId]                現在選択されている Face ID です。

### **solution.selected\_nodes()**

Returns                List[int]                現在選択されているノードのノード番号です。

## **View**

### **mw.display()**

ディスプレイをリフレッシュします。

### **mw.set\_active\_configuration(name)**

アクティブ・コンフィグレーション (active\_configuration) を変更します。

name                    str

### **mw.set\_model\_active()**

ディスプレイの更新後、現在のモデルを表示します。

## **Mesh tools**

### **mw.all\_elements()**

Returns                List[int]                メッシュのすべての要素番号です。

### **mw.all\_nodes()**

Returns                List[int]                メッシュ中のすべての要素番号です。

### **mw.automesh\_2d(element\_ids, max\_element\_size=0, grading=0.3, refinements=None)**

element_ids	list	
max_element_size	float	0 は最大制限なし
grading	float	サイズ累進 0.1(徐々)から 1.0(積極的)
refinements	List[tuple[int, float]] or None	ノード精製 リスト中の各精製は、精製要因が後続する境界ノード番号の 2 タプルです。

### **mw.automesh\_3d(element\_ids, volume\_mesh=True, max\_element\_size=0, min\_element\_size=0, refinements=None)**

element_ids	list	
volume_mesh	bool	
max_element_size	float	0 は最大制限なし
min_element_size	float	
refinements	List[tuple[int, float]] or None	ポイント精製 リスト中の各精製は、精製要因が後続する境界ノード番号の 3 タプルです。

### **mw.delete\_elements(element\_ids)**

指定要素を削除します。この方法は要素を再番号付します。

element_ids	list
-------------	------

### **mw.delete\_nodes(node\_ids)**

指定ノードとその関連要素を削除します。この方法はノードの再番号付し、要素が削除される場合は要素を再番号付します。

node_ids	list
----------	------

### **mw.element\_shape(element\_id)**

element_id	int	
Returns	str	シェープ : "line2", "line3", "tri3", "tri6", "quad4", "quad8", "tet4", "tet10", "pyr5", "pyr13", "wedge6", "wedge15", "hex8", "hex20"

### **mw.face\_nodes(face\_id)**

face_id	FaceId	
Returns	List[int]	フェイスのノード番号

### **mw.new\_element(shape, node\_ids, component=None)**

新しい要素を作成します。

shape	str	それらは、 "line2", "line3", "tri3", "tri6", "quad4", "quad8", "tet4", "tet10", "pyr5", "pyr13", "wedge6", "wedge15", "hex8", "hex20"のいずれかです。
node_ids	list	
component	str	
Returns	int	新しく作成された要素の要素番号を返します。

### **mw.new\_node(position)**

新しい節点を作成します。

position	Vector	
Returns	int	新しく作成されたノードのノード番号を返します。

### **mw.node(node\_id)**

node_id	int	
Returns	Vector	ノードの位置を返します。

### **mw.nodes(element\_id)**

element_id	int	
Returns	List[int]	指定要素番号のノード番号をリストします。

### **mw.refine\_custom(element\_ids, r, s, t)**

element_ids	list	
r	int	各要素の R 成分の番号を返します。
s	int	各要素の S 成分の番号を返します。
t	int	各要素の T 成分の番号を返します。

### **mw.refine\_x2(node\_ids=None, face\_ids=None, element\_ids=None)**

node_ids	list	ノード番号を返します。
face_ids	list	フェイス番号を返します。
element_ids	list	要素番号を返します。

### **mw.refine\_x3(node\_ids)**

node_ids	list	ノード番号を返します。
----------	------	-------------

### **mw.set\_node\_x(node\_id, value)**

ノードの X 座標を設定します。

node_id	int
value	float

### **mw.set\_node\_y(node\_id, value)**

ノードの Y 座標を設定します。

node_id	int
value	float

### **mw.set\_node\_z(node\_id, value)**

ノードの Z 座標を設定します。

node_id	int
value	float

### **mw.smooth(element\_ids)**

element_ids	list
-------------	------

### **mw.unrefine\_x2(element\_ids)**

element_ids	list
-------------	------

### **solution.all\_elements()**

Returns            List[int]            メッシュのすべての要素番号を返します。

### **solution.all\_nodes()**

Returns            List[int]            メッシュのすべてのノード番号を返します。

### **solution.element\_shape(element\_id)**

新しい要素を作成します。

element\_id            int

Returns            str            それらは、 "line2", "line3", "tri3", "tri6", "quad4", "quad8", "tet4", "tet10", "pyr5", "pyr13", "wedge6", "wedge15", "hex8", "hex20" のいずれかです。

### **solution.face\_nodes(face\_id)**

face\_id            FaceId

Returns            List[int]            face の[int]ノード番号をリストします。

### **solution.node(node\_id)**

node\_id            int

Returns            Vector            ノード位置を返します。

### **solution.nodes(element\_id)**

element\_id            int

Returns            List[int]            要素ノード ID の要素ノード番号です。

## **Solution**

### **mw.set\_solution\_active()**

ディスプレイの更新後、現在の解析モデルが表示されるようにします。

### **mw.solve()**

アクティブ・コンフィグレーション (active\_configuration) を解析します。ソルバーは同期して実行されますが、解析が終了するまで反応することなく、また進捗情報も示されません。

### **solution.buckling\_factor(mode)**

mode                    int

Returns                float                    指定モード番号のバックリング係数を返します。

### **solution.element\_node\_value(variable, element\_id, element\_node\_id, time\_step=None, mode=None)**

variable                str                    名前は、.liml ファイル中の<elementnodevariable>で使ったのと同じです。

element\_id             int

element\_node\_id       int

time\_step              int or None            時間ステップが存在する場合の時間ステップ番号です。

mode                    int or None            モードが存在する場合のモード番号です。

Returns                float or None          指定要素の要素ノードの解析変数値を返します。解析変数値がない場合は、None を返します。

### **solution.frequency(mode)**

mode                    int

Returns                float                    指定モード番号の周波数を返します。

### **solution.interpolate(variable,point,time\_step=None,mode=None)**

variable                str                    \*.lim の<nodevaliable>要素で使った名前などの変数名

point                   Vector                  非メッシュ・ポイント

time\_step              int or None            解析データ等の time\_step

mode                    int or None            解析データ等のモード数

Returns                float or None          解析値または、メッシュ・ポイント

### **solution.modes()**

Returns                List[int]               解析結果のモード番号を返します。



### **solution.node\_value(variable, node\_id, time\_step=None, mode=None)**

variable	str	variable は変数の名前です。名前は、.liml ファイル中の<nodevariable>で要素に使用したのと同じです。
node_id	int	
time_step	int or None	解析結果に時間ステップが存在する場合、時間ステップ数であり、存在しなければ、None です。
Mode	int or None	解析結果にモードが存在する場合、モード番号であり、存在しなければ、None です。
Returns	float or None	指定ノードの解析結果変数値であり、それが変数値を持たない場合は、None です。

### **solution.set\_variable(variable, values, time\_step=None, mode=None)**

ノード値の解析変数を返します。同じ名前の変数が既に存在する場合、上書きされるか、または生成され、任意の定式も、定式変数の値を変更することができないように新しい値として、再定義されます。

variable	str	変数の名前。無効の場合は処理の前に削除されます。名前は、.liml ファイル中の<nodevariable>要素で使用したのと同じ物です。
values	list	各ノードの変数値。リストの長さは、解析のノード番号と等しいに違いありません。インデックス 0 の値はノード 1 などに相当します。それは、値のないノードを含むことができません。
time_step	int or None	時間が存在する場合、時間ステップ数です。またはそうでないかもしれません。
mode	int or None	解析またはモードが他の方法でないモード番号
Returns	str	解析変数の実際の名前

### **solution.surface\_integral(variable, face\_id, time\_step=None, mode=None, vector=Fales)**

指定範囲の要素 Face の総面積を返します。

Variable	str	variable は変数の名前です。名前は、.liml ファイル中の<nodevariable>で要素に使用したのと同じです。
face_Id	FaceId	

time_step	int or None	解析結果に時間ステップが存在する場合、時間ステップ数であり、存在しなければ、None です。
mode	int or None	解析結果にモードが存在する場合、モード番号であり、存在しなければ、None です。
vector	bool	スカラ・フィールドも統合を回避します。ベクトル・フィールドの正規コンポーネントを統合する場合は真です。ベクトル・フィールドは、3つのコンポーネントが使用されますが、「変数」値は大きさ変数の名前であればなりません。
Returns	float	

### **solution.time(time\_step)**

time\_step int

Returns	float	指定時間ステップ数です。
---------	-------	--------------

### **solution.time\_steps()**

Returns	List[int]	解析結果に存在するステップ数です。
---------	-----------	-------------------

### **solution.volume\_integral(variable, element\_Id, time\_step=None, mode=None)**

指定範囲の要素の総体積を返します。

Variable	str	variable は変数の名前です。名前は、.liml ファイル中の<nodevariable>で要素に使用したのと同じです。
element_id	int	
time_step	int or None	解析結果に時間ステップが存在する場合、時間ステップ数であり、存在しなければ、None です。
Mode	int or None	解析結果にモードが存在する場合、モード番号であり、存在しなければ、None です。
Returns	float	

## Tools

### **mw.centroid(element\_id)**

element\_id          int

Returns              Vector              指定要素の中心位置です。

### **mw.mass(element\_id)**

element\_id          int

Returns              float              指定要素の質量（Mass）を返します。

### **mw.surface\_area(face\_id)**

face\_id              FaceId

Returns              float              指定要素 Face の面積を返します。

### **mw.volume(element\_id)**

element\_id          int

Returns              float              指定要素の体積を返します。

### **solution.centroid(element\_id)**

element\_id          int

Returns              Vector              指定要素の中心位置を返します。

### **solution.mass(element\_id)**

element\_id          int

Returns              float              指定要素の質量を返します。

### **solution.surface\_area(face\_id)**

face\_id              FaceId

Returns              float              指定要素 Face の面積です。

### **solution.volume(element\_id)**

element\_id          int

Returns              float              指定要素の体積です。

## **Help**

### **mw.version()**

Returns              int              Mecway のバージョン番号を返します。

## **Geometry**

### **mw.generate\_mesh()**

STEP ファイルのすべてのメッシュを生成します。

## **Components & Materials**

### **mw.component\_elements(component)**

component          str              コンポーネント名

Returns              List[int]              コンポーネントに含まれる要素番号

### **mw.components()**

Returns              List[str]              コンポーネント名

### **mw.delete\_component(name)**

コンポーネントを削除します。

Name                  str              コンポーネント名

### **mw.delete\_material(material)**

材料特性の削除

material              str              材料特性名

### **mw.element\_component(element\_id)**

element\_id          int

Returns	str	要素に含まれるコンポーネント名
---------	-----	-----------------

### **mw.material(component)**

component	str	コンポーネント名
-----------	-----	----------

Returns	str or None	指定のコンポーネントの材料特性名
---------	-------------	------------------

### **mw.material\_property(material, property)**

material	str	材料特性です。
----------	-----	---------

property	str	材料特性名です。多くの材料特性は、.liml ファイルの<mat>要素の属性名と同じです。ラミネート・レイヤーデータでは、「layer」です。また、カーブ・データを強固にするプラスチックでは、「hardeningcurve」です。
----------	-----	---

Returns	float or List[List[float]]	材料特性の値です。ラミネートおよび温度依存する特性のような平板状のデータではテーブルの列リストです。
---------	----------------------------	--

### **mw.material\_type(material, group)**

material	str	材料特性名です。
----------	-----	----------

group	str	材料グループ名です。それらは、"geometric", "mechanical", "density", "plastic", "thermal", "fluid", "em", "failure", "piezoelectric"のいずれかです。
-------	-----	--

Returns	str	指定グループの材料タイプ名です。
---------	-----	------------------

### **mw.new\_component(name=None)**

新しいコンポーネントを作成します。

name	str or None	新しいコンポーネント名です。それが既に存在するか無効の特徴を含む場合、修正されます。自動的に名前を生成するには、これをセットしないでください。
------	-------------	---

Returns	str	作成されたコンポーネント名です。これは望む名前とは異なるかもしれません。
---------	-----	--------------------------------------

### **mw.new\_material(name=None)**

新しい材料を作成します。

name	str or None	新しい材料名です。それが既に存在するか無効の特徴を含む場合、修正されます。自動的に名前を生成するには、これをセットしないでください。
------	-------------	--

Returns	str	作成された材料名です。これは望む名前とは異なるかもしれません。
---------	-----	---------------------------------

### **mw.rename\_material(material, name)**

材料名を変更します。

material	str	既存の材料名です。
----------	-----	-----------

name	str	新しい名前です。別の材料がこの名前を持っているか無効の特徴を含む場合、それは修正されます。
------	-----	---

Returns	str	新しい名前です。これは既存の名前とは異なるかもしれません。
---------	-----	-------------------------------

### **mw.set\_element\_component(element\_id, component)**

コンポーネントに要素を指定します。

element_Id	int
------------	-----

component	str	コンポーネント名です。
-----------	-----	-------------

### **mw.set\_material(component, material)**

コンポーネントに既存の材料を指定します。

component	str	コンポーネント名です。
-----------	-----	-------------

material	str or None	材料名です。それを削除せず、コンポーネントから材料を取り除くには、これをセットしないでください。
----------	-------------	--

### **mw.set\_material\_property(material, property, value)**

material	str	材料名です。
----------	-----	--------

property	str	多くの材料特性は、.liml ファイルの<mat>要素の属性名と同じです。ラミネート・レイヤーデータでは、「layer」です。また、カーブ・
----------	-----	--

データを強固にするプラスチックでは、「hardeningcurve」です。

value	float or list	材料特性値です。ラミネートや時間依存のデータはテーブル値として設定します。
-------	---------------	---------------------------------------

### **mw.set\_material\_type(material, group, type)**

material	str	材料名です。
property	str	材料グループ名です。多くの材料特性は、.liml ファイルの<mat>要素の属性名と同じです。ラミネート・レイヤーデータでは、「layer」です。また、カーブ・データを強固にするプラスチックでは、「hardeningcurve」です。
type	str	指定グループの材料特性タイプ名です。これは、.liml ファイルの<mat>要素の子要素の”type”値と同じです。

### **solution.component\_elements(component)**

component	int	コンポーネント名です。
Returns	list[int]	コンポーネントに付属する要素番号です。

### **solution.components()**

Returns	list[str]	コンポーネント名です。
---------	-----------	-------------

### **solution.elements\_component(element\_id)**

element_id	int	
Returns	str	要素に付属するコンポーネント名です。

### **solution.material (component)**

component	str	コンポーネント名です。
Returns	str or None	指定コンポーネントの材料特性名または未定

### **solution.material\_property (material, property)**

material	str	材料名です。
property	str	材料特性名です。多くの材料特性は、.liml ファイルの<mat>要素の属性

名と同じです。ラミネート・レイヤーデータでは、「layer」です。また、カーブ・データを強固にするプラスチックでは、「hardeningcurve」です。

Returns float or List[List[float]] 材料特性値です。ラミネートまたは温度依存データはテーブル・データです。

### **solution.material\_type(material, group)**

material str 材料名です。

group str 材料グループ名です。それらは、"geometric", "mechanical", "density", "plastic", "thermal", "fluId", "em", "failure", "piezoelectric"のいずれかです。

Returns float グループに付属する材料タイプを返します。

## **Loads & Constraints**

### **mw.load\_property(load, property)**

解析特性や拘束条件の値を得ます。それらは拘束条件、対称条件、節点温度、節点内部熱などです。

load str 荷重名や拘束名です。

property str 特性名です。これは、.liml の特性要素名です。

Returns float or List[List[float]] 特性値です。それは数値やテーブル値ですが、式は含まれません。テーブル値は、1 列または 2 列のデータです。

### **mw.set\_load\_property(load, property, value)**

load str 荷重名や拘束名です。

property str 特性名です。これは、.liml ファイルの特性要素名と同一です。

value float or List 特性値です。それは数値やテーブル値ですが、式は含まれません。テーブル値は、1 列または 2 列のデータです。

## **Named selections**

### **mw.add\_to\_named\_selection(name, item)**

name str

item int or FaceId ノード番号、フェイス Id、name 選択タイプの要素番号



### **mw.delete\_named\_selection(name)**

name                      str

### **mw.named\_selection(name)**

name                      str

Returns                  List[int] or List[FaceId]    ノード番号、フェイス Id、name 選択タイプの要素番号

### **mw.named\_selection()**

Returns                  List[str]                      選択名

### **mw.new\_element\_selection(name=None)**

name                      str                      選択された名前の矛盾を修正あるいは削除します。

Returns                  str                      選択された名前

### **mw.new\_face\_selection(name=None)**

name                      str                      選択された名前の矛盾を修正あるいは削除します。

Returns                  str                      選択された名前

### **mw.new\_node\_selection(name=None)**

name                      str                      選択された名前の矛盾を修正あるいは削除します。

Returnstr                str                      選択された名前

### **mw.new\_remove\_named\_selection(name, item)**

name                      str

Returns                  str                      ノード番号、フェイス Id、選択名の要素番号

### **mw.new\_rename\_named\_selection(name, new\_name)**

name                      str                      選択された名前

new\_name                str                      新しい選択された名前。選択された名前の矛盾を修正あるいは削除します。

Returns                  str                      新しい選択された名前

### **solution.named\_selection(name)**

name                      str

Returns                  List[int] or List[FaceId]    ノード番号、要素番号、選択名のフェイス Id

### **solution.named \_selection()**

Returns                List[str]                選択名

## **Extra**

### **mw.input(prompt)**

テキストボックスを備えたポップアップ・ダイアログを表示し、ユーザーがそれを閉じるまで待ちます。

prompt                str

Returns                str                ユーザーが入力したテキスト文字 (str タイプ) を返します。

### **mw.message(message)**

ポップアップ・ダイアログのメッセージを表示し、ユーザーがそれを閉じるまで待ちます。

message                str

## **Classes**

### **FaceId(element\_id, face\_number)**

以下の特性を備えたオブジェクトを返します。

element\_id            int

face\_number           int                要素の Face ID 1-6 の番号です。その値は、.liml ファイル中の<face>要素で  
使用したものと同一です。

### **Vector(x, y, z)**

以下の特性を備えたオブジェクトを返します。

x                      float

y                      float

z                      float

## 例 (Samples)

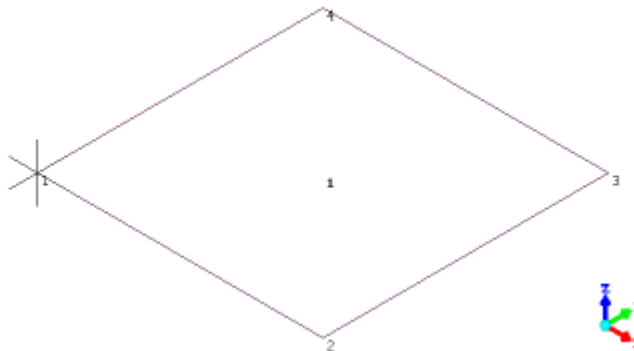
### generate\_mesh.py

1つの Quad4 要素 (1×1mm) を定義します。このスクリプトは Mecway13 の Samples に紹介されています。

```
# This example script creates an element and its nodes, similar to
#
#           Mesh tools -> Create -> Quick square
#
assert mw.version() == 13
node_1 = mw.new_node(Vector(0, 0, 0))
node_2 = mw.new_node(Vector(1, 0, 0))
node_3 = mw.new_node(Vector(1, 1, 0))
node_4 = mw.new_node(Vector(0, 1, 0))
mw.new_element("quad4", [node_1, node_2, node_3, node_4])
```

### 操作方法

- Tools > Scripts... を実行し、New ボタンをクリックし、../generate\_mesh.py を選択します。
- Tools メニューに登録された generate\_mesh.py を実行します。



### 備考

- この例では、単純な 1 要素のプレートが作成されます。解析を進めるためには、Material 特性を定義しなければなりません。また、必要に応じて、Loads&Constraints から荷重や境界条件を定義してください。
- 例えば、Refine×2 を実行し、Material 特性の Geometric として、Shell / membrane を選択し、Thickness=0.01m を設定します。また、Mechanical として、Young's modulus=100GPa、Poisson's=0.3 を設定します。
- さらに、境界条件として、側辺を fixed support (拘束) し、また、荷重として、Loads&Constraints > New pressure から全サーフェイスに対し、Normal 成分に pressure=10 kPa を設定します。

## transfer\_displacements\_from\_solution.py

解析結果の節点変位 (displx,disply,displz) をオリジナルの座標値に加算します。このスクリプトは、Mecway13 の Samples に紹介されています。

```
# This example script replicates the functionality of
#
#           Mesh tools -> Transfer displacements from solution
#
# but is limited to solutions without time steps or modes and lacks error
# checking.
```

```
assert mw.version() == 13
```

```
for node_Id in solution.all_nodes():
```

```
    # Read displacement from solution
    displacement_x = solution.node_value("displx", node_Id)
    displacement_y = solution.node_value("disply", node_Id)
    displacement_z = solution.node_value("displz", node_Id)
```

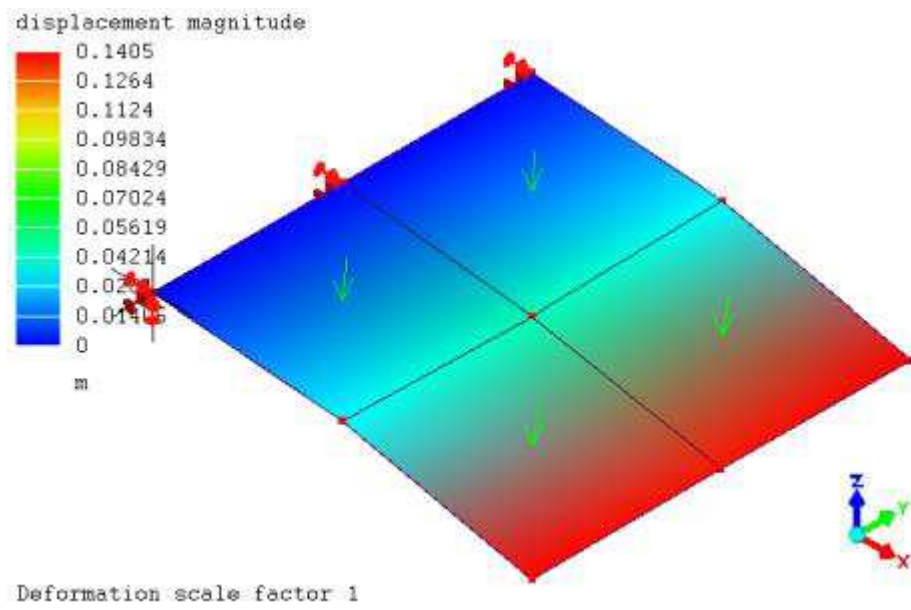
```
    # Calculate new node position
    new_x = mw.node(node_Id).x + displacement_x
    new_y = mw.node(node_Id).y + displacement_y
    new_z = mw.node(node_Id).z + displacement_z
```

```
    # Change node position
    mw.set_node_x(node_Id, new_x)
    mw.set_node_y(node_Id, new_y)
    mw.set_node_z(node_Id, new_z)
```

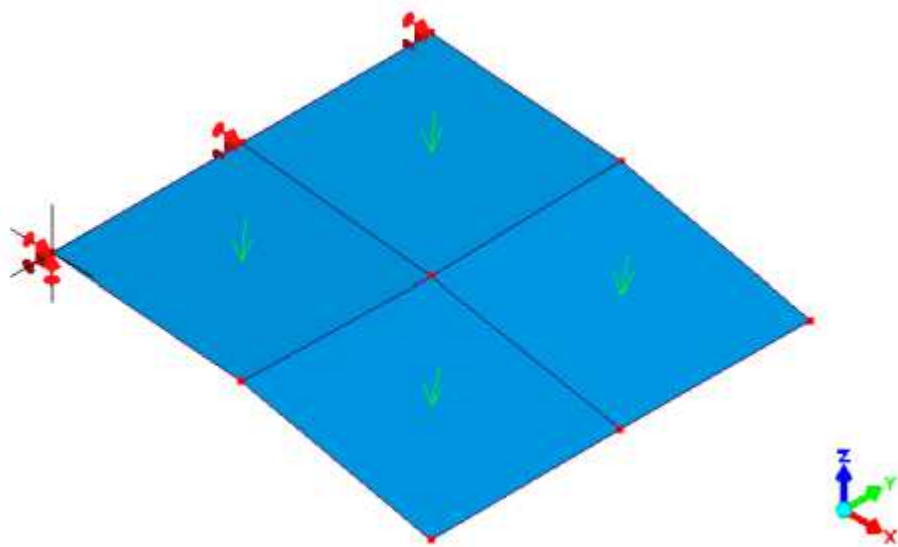
```
mw.message("Finished")
```

### 操作方法

- 前例の generate\_mesh.py および備考の解析条件の設定し、解析の実行後、Tools>Scripts... から Script ダイアログを開き、New ボタンをクリックし、../transfer\_displacements\_from\_solution.py を選択します。
- Tools メニューに登録された transfer\_displacements\_from\_solution.py を実行します。



静解析（pressure 荷重）結果の変形シェープ



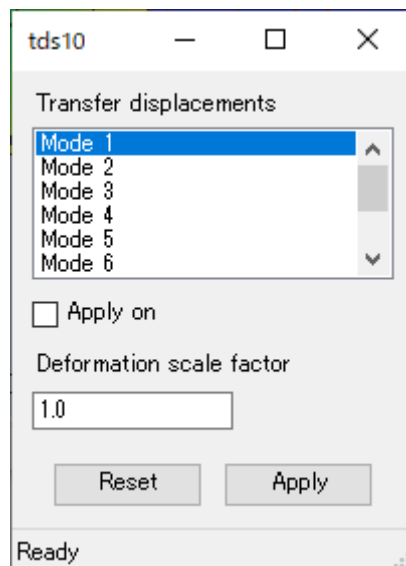
オリジナル・モデルの変形シェープ加算モデル

## 参考例

### メッシュ・ツール

#### tds10

解析結果の節点座標変位値 (displx,disply,displz,...) をオリジナルの座標値に加算します。このスクリプトは、Samples > transfer\_displacements\_from\_solution.py の応用プログラムです。



tds10 ダイアログ

- Transfer displacement リストボックス  
以下の解析結果タイプのサポートデータ・アイテムを選択し、Apply ボタンをクリックします。
  - 静的変位シェープ (solution.displacement)
  - モード変位シェープ (solution.mode)
  - ダイナミック・シェープ (solution.time\_step)
- Apply on チェックボックス  
Transfer displacement リストボックスのアイテムの選択時に更新 (Apply) が実行されます。
- Deformation scale factor テキストボックス  
解析結果の変形スケールを指定します。
- Reset ボタンをクリックするとオリジナルの節点座標に戻すことができます。

#### 備考

- スケール係数 (Deformation scale factor) を変更し、更新 (Apply) を実行すると、そのスケール係数に基づく変形座標モデルが表示されます。
- 実行後 (Close) に最後の更新 (Apply) に基づいて、オリジナルの節点座標が更新されます。

## refine10.py

Mecway メッシュを指定の分割数、バイアス (Bias) にしたがって、メッシュ分割 (refine) します。

Elem. axis	No. of Division	Bias (-10 to 10)
R	1	0
S	1	0
T	1	0

refine10 ダイアログ

- 要素タイプごとの要素軸 (Elem. axis) の分割数 (No. of Division) とバイアス (Bias) にしたがって要素が分割されます。
  - Line2 要素は R 軸の分割数とバイアスの設定が使用されます。
  - Quad4 要素は R 軸と S 軸 の分割数とバイアスの設定が使用されます。
  - Wedge6 要素は T 軸の分割数とバイアスの設定が使用されます。
  - Hexa8 要素は R 軸、S 軸、T 軸の分割数とバイアスの設定が使用されます。
- バイアス (Bias) 値は範囲-10~10 で設定することが推奨され、次のように解釈されます。
  - 0 : 各要素は等間隔で分割されます。
  - 0 以上 (1,2,...) : 各要素は小間隔から大間隔 (昇順間隔) に分割されます。
  - 0 以下 (-1,-2,...) : 各要素は大間隔から小間隔 (降順間隔) に分割されます。

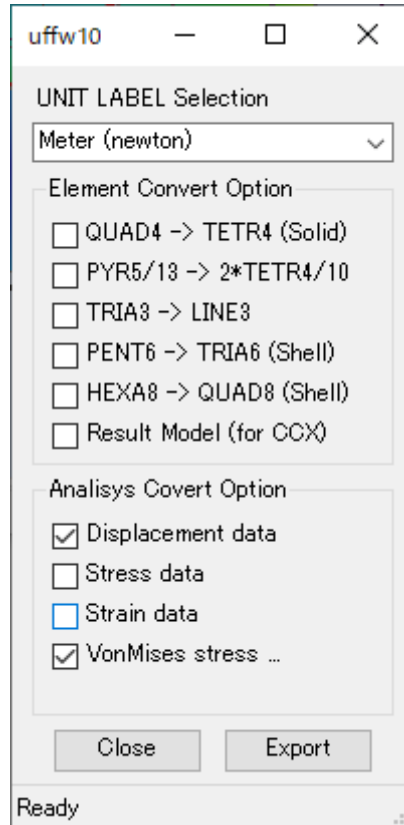
## 備考

- Mecway モデルの分割に伴って、新たな節点が生成され、重複節点が生じるため、refine 実行後 (Close ボタンの実行後)、節点のマージ処理を実行しなければなりません。これは現状 (Mecway13) の Python インタプリタ (mw 関数) では、節点マージ・コマンドがサポートされていないため、後処理として必要になります。
- Mecway モデルの分割に伴って、オリジナルの節点番号、要素番号が更新されます。
- Mecway モデルの分割に伴って、オリジナルのコンポーネント構成が更新されます。

## UFF インターフェイス

### uffw10.py

Mecway モデル、解析結果をユニバーサル・ファイル・フォーマット（拡張子：uff、unv）の FEM データとして、エクスポートします。



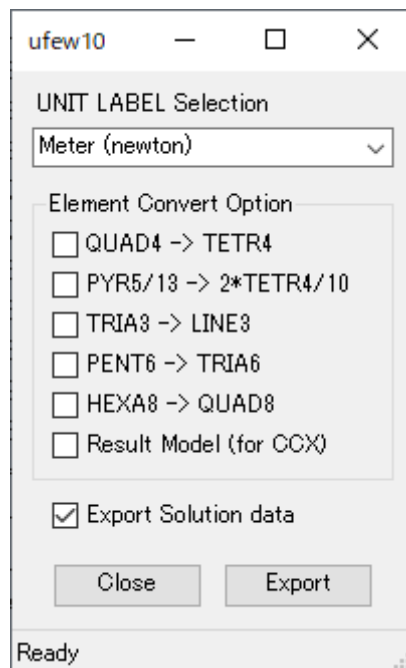
uffw10 ダイアログ

- UNIT LABEL Slection : Mecway モデルの単位（長さ）ラベルを選択します。この設定は単位ラベル情報として、UFF データセット 164 に設定されます。
- Element Convert Option : 要素タイプは節点数によって識別されるため、節点数ごとに 2 者択一で Mecway モデルの要素タイプなどを選択します。
  - 4 節点要素の場合、QUAD4（デフォルト）または TETR4
  - 5/13 節点要素の場合、PYR5/13（デフォルト）または 2\*TETR4/10
  - 3 節点要素の場合、TRIA3（デフォルト）または LINE2
  - 6 節点要素の場合、PENT6（デフォルト）または TRIA6
  - 8 節点要素の場合、HEXA8（デフォルト）または QUAD8
  - CCX ソルバーによる結果（SHELL->SOLID 要素コンバートモデル）を使用します。
- Analysis Convert Option : 変換対象データを選択します。
  - Displacement data（デフォルト：オン）
  - Stress data
  - Strain data
  - VonMises stress data



## ufew10.py

Mecway モデル、解析結果をユニバーサル・ファイル・フォーマット（拡張子：uff、unv）の TEST データとして、エクスポートします。



ufew10 ダイアログ

- UNIT LABEL Slection : Mecway モデルの単位（長さ）ラベルを選択します。その設定は単位ラベル情報として、UFF データセット 164 に設定されます。
- Element Convert Option : 要素タイプは節点数によって識別されるため、節点数ごとに 2 者択一で Mecway モデルの要素タイプなどを選択します。
  - 4 節点要素の場合、QUAD4（デフォルト）または TETR4
  - 5/13 節点要素の場合、PYR5/13（デフォルト）または 2\*TETR4/10
  - 3 節点要素の場合、TRIA3（デフォルト）または LINE3
  - 6 節点要素の場合、PENT6（デフォルト）または TRIA6
  - 8 節点要素の場合、HEXA8（デフォルト）または QUAD8
  - CCX ソルバーによる結果（SHELL->SOLID 要素コンバートモデル）を使用します。
- Export Solution data : 解析結果データが存在し、その解析結果データをエクスポートする場合に選択します。（デフォルト：オン）