

### ■ タイムステップ番号が不連続なデータ

FieldViewで非定常データを扱う場合、ファイル名は連番であることが必要です。

FVXではこの連番をタイムステップ番号として扱いますが、対象となるファイルの連番が連続していない場合、たとえば、何らかの事情で1の次に20が来て、その次に100が来るなどというタイムステップ番号が不連続となる解析結果の可視化をする必要がある場合、タイムステップが単調増加するような単純なループで対応することが出来ません。

### ■ タイムステップ番号の取得とループ

このようなデータの場合、タイムステップ番号をリストの形で取得し、ループはその要素の数を終端として実行することにします。ループの中で、`set_transient`でデータを呼び出す場合は、ループカウンタによる間接参照を行います。

## ■ プログラムコード：データ読込部

```
file_name="rect_duct_001.uns"

data_input_table = {
    data_format = "unstructured",
    input_parameters = {
        name = file_name,
        options = {
            input_mode = "replace",
            transient = "on"
        }
    }
}

local handle=read_dataset(data_input_table)
```

- プログラムの解説  
データ読み込み部分です。  
ここでは、FieldViewのサンプルデータである  
Rectangular Ductのデータを連番が不連続になるようにコピーして、それらを非定常データとして読み込ませます。

連番ファイル：

rect\_duct\_001.uns  
rect\_duct\_002.uns  
rect\_duct\_010.uns  
rect\_duct\_014.uns  
rect\_duct\_020.uns  
rect\_duct\_037.uns

連番は飛び飛びですが、これらのデータでも非定常データとして読み込むことは可能です。

## ✓ query\_transient

非定常データについての情報を取得するときに使用する関数です。

実行すると以下の値が得られます。

```
local ts=query_transient()
```

## transient\_table

このテーブルは、query\_transient関数からの出力を表しています。

Field	Data Type	Comments	Default
time_step	number	Current time step.	n/a
solution_time	number	Current solution time.	n/a
total_time_steps	number	time_stepの数 Read Only.	n/a
time_step_range	table	Read Only.	n/a
min	number	time step numberの最小値	n/a
max	number	time step numberの最大値	n/a
values	table	time step numberのテーブル total_time_stepの数だけエントリがあります。	n/a
solution_time_range	table	Read Only.	n/a
min	number	solution timeの最小値	n/a
max	number	solution timeの最大値	n/a
values	table	solution time stepの値の格納されているテーブル total_time_stepの数だけエントリがあります。	n/a

ここで、time\_step\_rangeの valuesに、タイムステップ番号の一覧が格納されているので、これを使ってたとえば飛び飛び（不連続）であったとしても、タイムステップ番号を取得することが可能です。

- ✓ Time Step番号の要素数だけ繰り返す  
getn関数を使って、time\_step\_range.valuesの要素数を取得し、ループの終端とします。

```
for i=1,getn(ts.time_step_range.values) do
```

- ✓ Set\_transient () によるタイムステップの更新  
ループカウンタ*i*を使ってts.time\_step\_range.values [i] とすれば、タイムステップ番号を取得できます。  
この番号を使って、set\_transientで当該タイムステップのデータを呼び出します。

```
local set_step = { time_step = ts.time_step_range.values[i]}  
set_transient(set_step)
```

あとは写真撮影などで、ビットマップ形式で可視化結果を出力します。

### サンプルコード : データ読込部

```
file_name="rect_duct_001.uns"

data_input_table = {
    data_format = "unstructured",
    input_parameters = {
        name = file_name,
        options = {
            input_mode = "replace",
            transient = "on"
        }
    }
}

local handle=read_dataset(data_input_table)
local ts=query_transient()
```

### サンプルコード : 主ループ

```
for i=1,getn(ts.time_step_range.values) do

    fv_script("RESTART ALL_NO_DATA_READ Step14-1.dat")

    --Time Stepを指定し、非定常データ呼び出し
    local set_step = { time_step = ts.time_step_range.values[i]      }
    set_transient(set_step)

    --写真撮影
    surface="surface"..tostring(ts.time_step_range.values[i])
    fv_script('PRINT GRAPHICS JPEG '..surface )

end
```