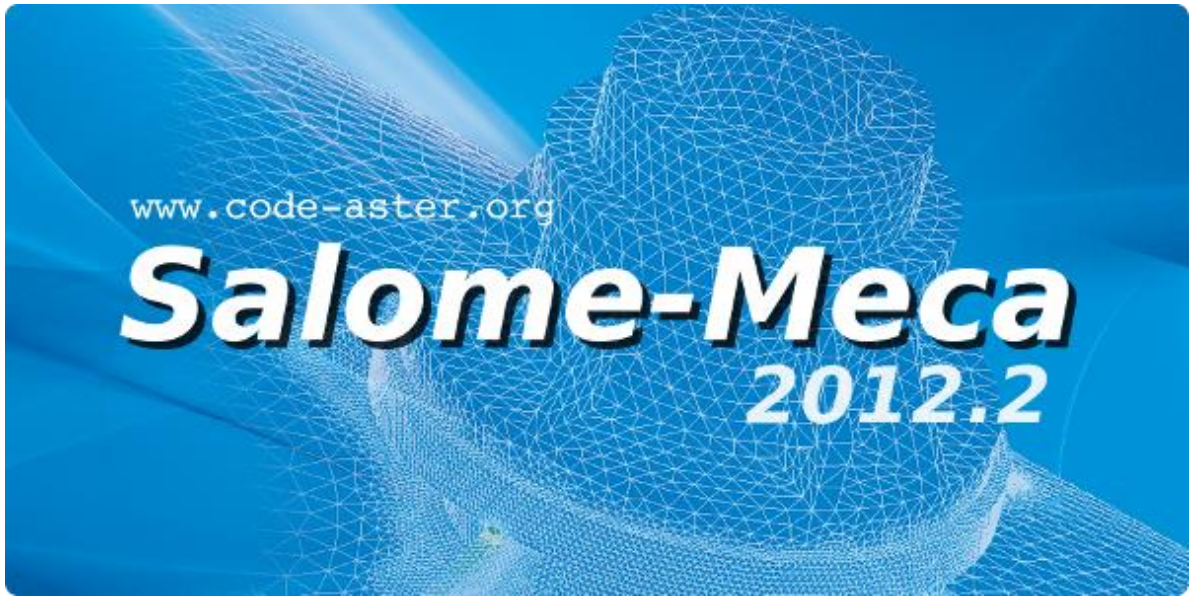
関数と数式



Code_Aster, Salome-Meca course material

定義

- **FONCTION**：
  1変数に依存するテーブル化（離散化）された関数
- **NAPPE**：
  2変数に依存するテーブル化（離散化）された関数
- **FORMULE**：
  複数の変数に依存する連続な数式

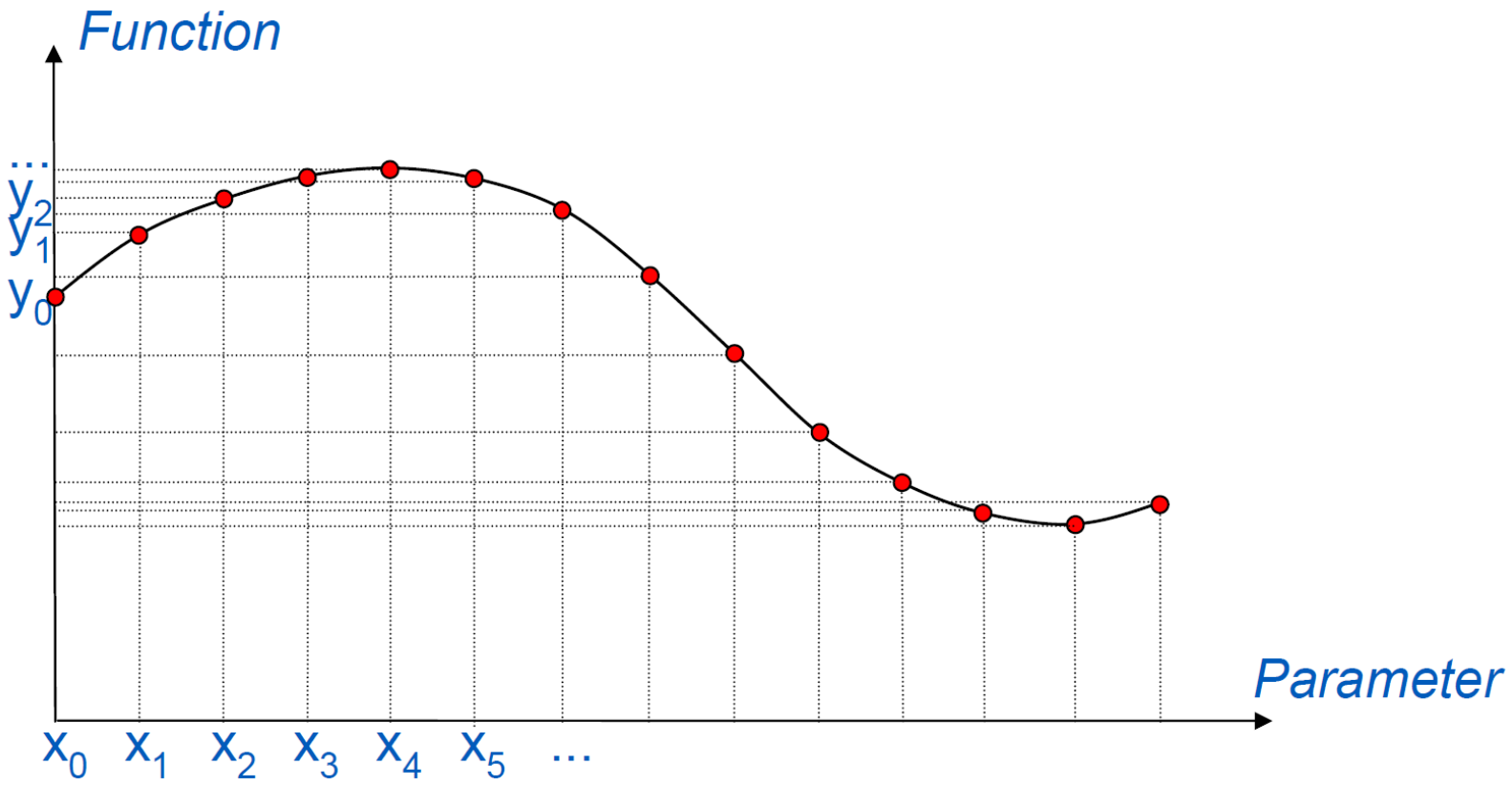
パラメータ

- *Code_Aster* でよく使われるパラメータ

ABSC_CURV	曲線の横軸	EPSI	ひずみ
DX	X方向変位	SIGM	応力
DY	Y方向変位		
DZ	Z方向変位		
DRX	X軸回りの回転	INST	時間
DRY	Y軸回りの回転		
DRZ	Z軸回りの回転		
X	X座標	TEMP	温度
Y	Y座標		
Z	Z座標		

コードとプラットフォームの一般原則

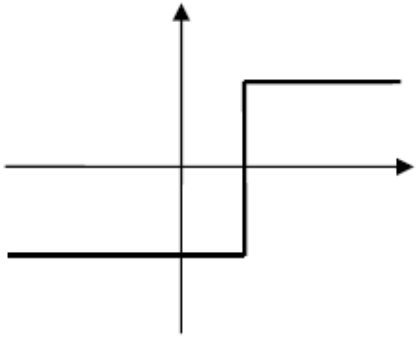
- 関数：(x_i, y_i) で定義



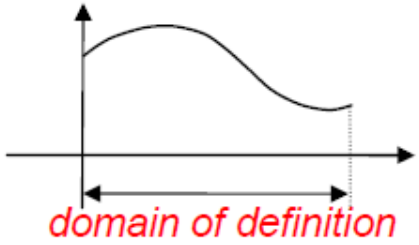
横軸の値は確実に増加させる：$x_0<x_1<x_2<x_3<x_4<x_5<x_6<x_7<x_8$

関数

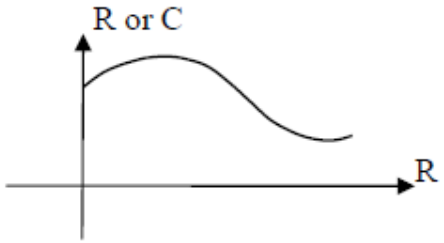
- **Code_Aster における関数：数学的意味**



1つの横軸値に対して複数の値
→ 関数でない

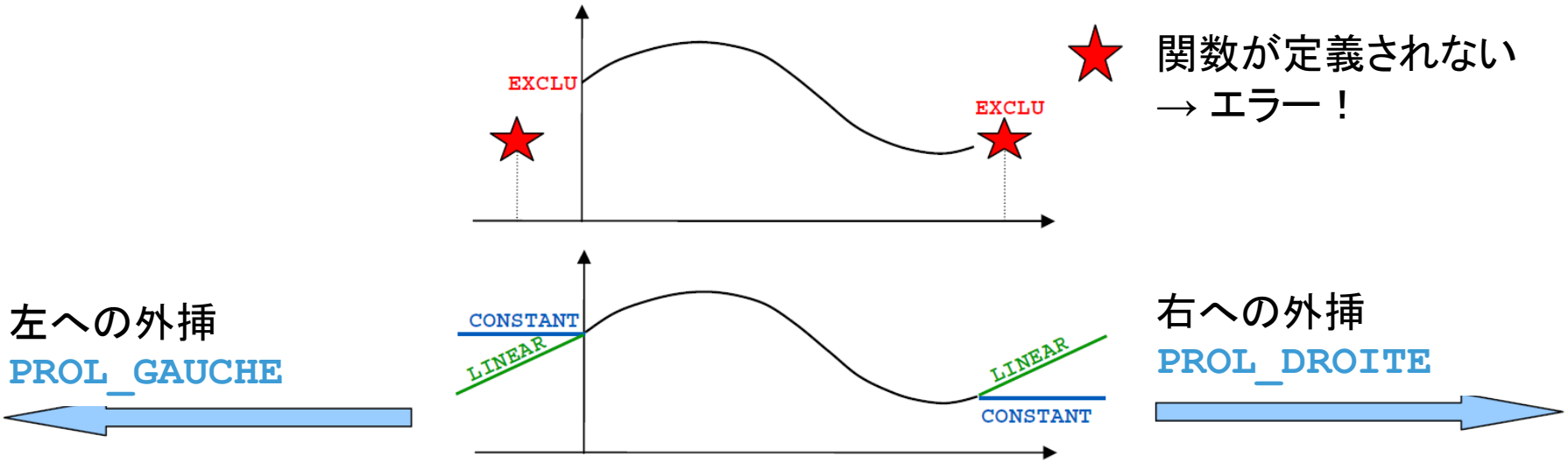


関数は定義域を持つ



関数は複素関数でも実数でもよいが，
パラメータは実数のみ

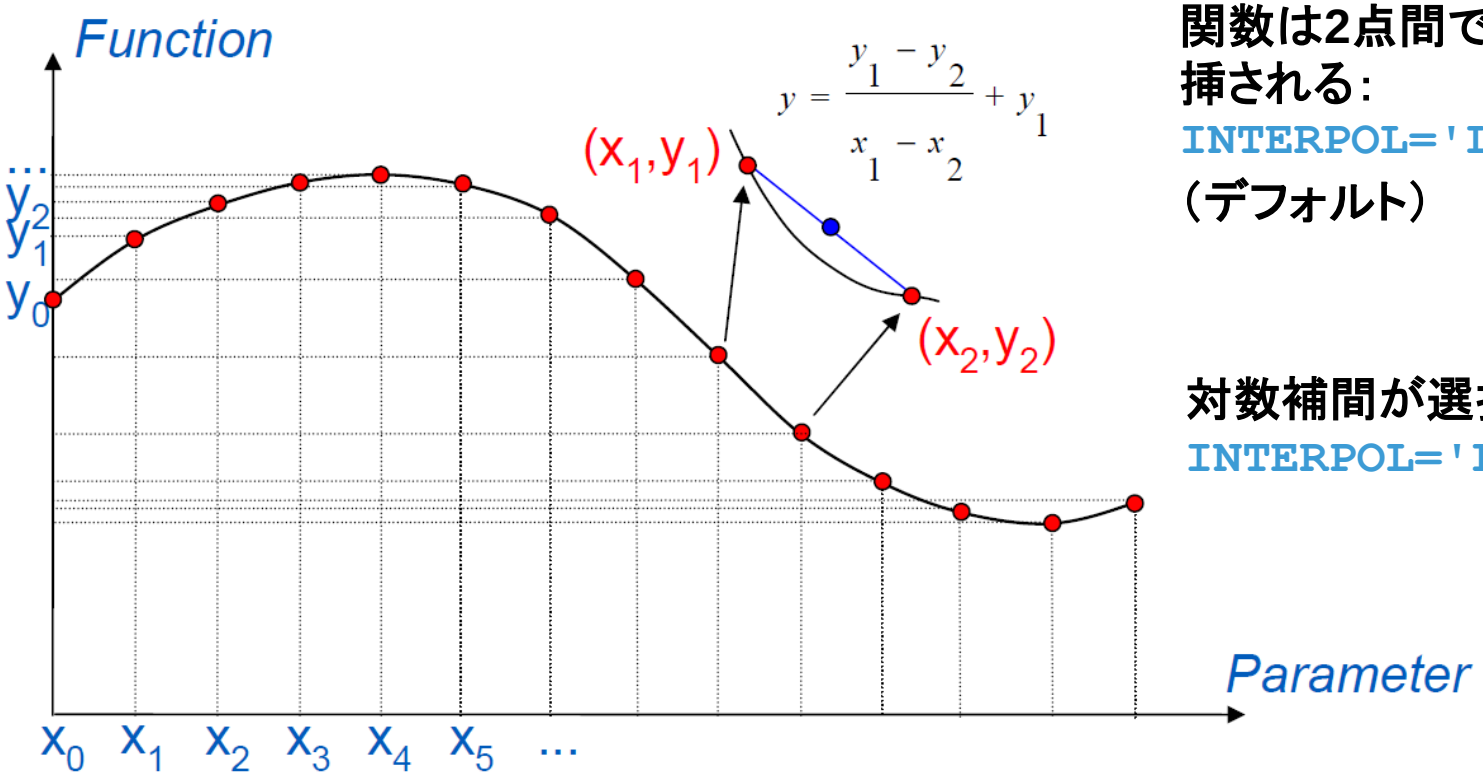
関数



EXCLU : 延長なし（デフォルト）
CONSTANT : 最終値のまま延長
LINEAIRE : 線形外挿

関数

関数:2点間の内挿



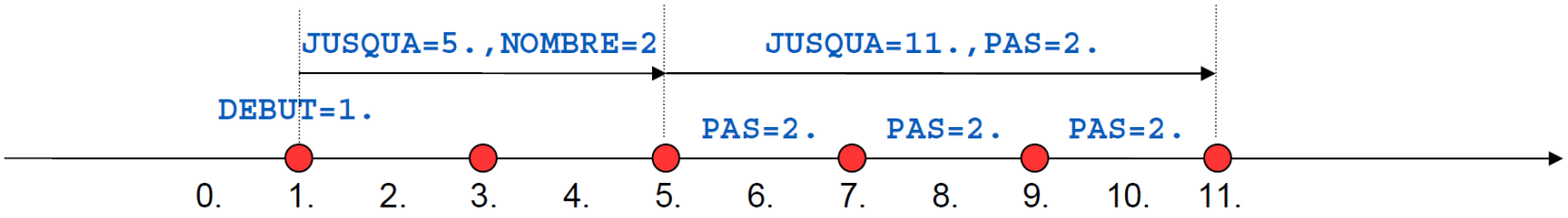
関数は2点間で線形内挿される:
`INTERPOL='LIN'`
(デフォルト)

対数補間が選択可:
`INTERPOL='LOG'`

実数値のリスト

- 実数値の定義:DEFI_LIST_REEL

```
ListR = DEFI_LIST_REEL(DEBUT      =1,.
INTERVALLE=(  _F(JUSQU_A=5.,NOMBRE=2,),
              _F(JUSQU_A=11.,PAS=2.,),))
```

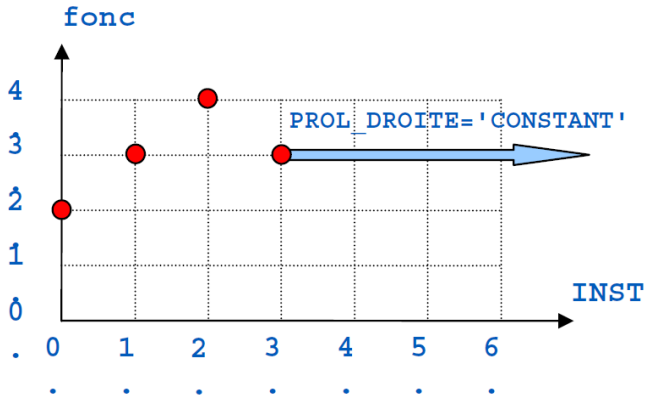



```
ListR = DEFI_LIST_REEL(VALE=(1.,3.,5.,7.,9.,11.))

ListR = DEFI_LIST_REEL(VALE=range(1.,13.,2.))
```

関数

関数:定義




```
fonc=DEFI_FONCTION(NOM_PARA='INST',
                   VALE=(   0. ,2. ,
                            1. ,3. ,
                            2. ,4. ,
                            3. ,3. ,
                   PROL_GAUCHE='EXCLU',
                   PROL_DROITE='CONSTANT')
```

複素関数:
VALEに替えてVALE_C

```
ABSC=DEFI_LIST_REEL(VALE=(1.,2.,3.,4.,))
ORDO=DEFI_LIST_REEL(VALE=(2.,3.,4.,3.,))
        fonc=DEFI_FONCTION(
                   NOM_PARA='INST',
                   VALE_PARA=ABSC,
                   VALE_FONC=ORDO,
                   PROL_GAUCHE='EXCLU',
                   PROL_DROITE='CONSTANT')
```

横軸は増加

関数

- 2変数関数：DEFI_NAPPE

```
nappe=DEFI_NAPPE(NOM_PARA='AMOR',
                 PARA=(0.01,0.02,),
                 FONCTION(DF1,DF2,)
                 PROL_GAUCHE='CONSTANT',
                 PROL_DROITE='CONSTANT')
```

関数の関数：
先行して定義された関数

```
nappe=DEFI_NAPPE(NOM_PARA='PULS',
                 PARA=(0.01,0.03,),
                 NOM_PARA_FONCT='INST',
                 DEFI_FONCTION=(
                 _F(VALE = (3.,1.,4.,2.,5.,3.,)),
                 _F(VALE = (3.,3.,4.,3.,5.,5.,))))
```

関数の関数：
DEFI_NAPPE で再定義

- 例として zzzz100a のテストケースを参照のこと

数式

- **数式：数学的な関数として定義**

```
form = FORMULE(NOM_PARA='X',
               VALE= '''sin(X)''' )
```

<<Pythonを計算機として使う>>
http://docs.python.org/tut/tut.html
http://docs.python.org/lib/module-math.html

Python math モジュールの主な関数が**デフォルトとして**
Code_Aster にインポートされている

三重引用符 ''' を用いてPython中の複数行にわたる数式を定義する

- **例として zzzz100a のテストケースを参照のこと**

数式

- Python関数としての数式

```
SIa = FORMULE(NOM_PARA='X',VALE='sin(X)')
X   = SIa(1.57)
print SIa(1.57)
```

Pythonで評価

```
SIa = FORMULE(NOM_PARA='X',VALE='sin(X)')
SIb = FORMULE(NOM_PARA='X',VALE(X*SIa(X)')
```

関数の関数

```
def = HEAVISIDE(x) :
      if x<0.    : return 0.
      if x>=0.   : return 1.
F_HVS = FORMULE( NOM_PARA = 'INST',
                 VALE     = 'HEAVISIDE(INST)')
```

高水準な定義の関数

$$HEAVISIDE(x) = \begin{cases} 0. & \text{if } x < 0. \\ 1. & \text{if } x \geq 0. \end{cases}$$

```
from math import pi
OMEGA = 30.
NAP = FORMULE(NOM_PARA = ('AMOR','FREQ'),
                  VALE =
      '''(1./((2.*pi*FREQ)**2 - OMEGA**2 )**2
         +(2.*AMOR*2.*pi*FREQ*OMEGA)**2)''')
```

複数の変数を持つ関数

- 例として zzzz100a のテストケースを参照のこと

数式

- FONCTION 中での FORMULE の変換: CALC_FONC_INTERP

```
SI     = FORMILE(NOM_PARA = 'INST',
                 VALE     = '''sin(INST)''')
DEPI         = 2.*pi
PAS0         = DEPI/200.
LI1          = DEFI_LIST_REEL(DEBUT =
0,INTERVALLE=_F(JUSQU_A=DEPI, PAS=PAS0),)
SI1          = CALC_FONC_INTERP(FONCTION    =SI,
                                LIST_PARA    = LI1,
                                PROL_GAUCHE ='EXCLU'
                                PROL_DROITE ='CONSTANT')
```

- NAPPE 中での FORMULE の変換: CALC_FONC_INTERP

- 例として zzzz100a のテストケースを参照のこと

注意

- **FONCTION と FORMULE の使い分け**
  - 関数はテーブル化される：低級な Fortran（たとえば，特性式）の中で使用される場合は速い
  - 数式は<<厳密>>：より高精度
  - **FORMULE** で定義して **CALC_FONC_INTERP** で使用する！

- **関数を使用する場面**
  - 境界条件（たとえば，**AFFE_CHAR_MECA_F** コマンド）
  - 特性式：弾塑性の引張曲線，温度依存性のある特性式
  - 非線形の境界条件に対する複合的な関数

その他のコマンド

ファイルから関数を読み込む	**LIRE_FONCTION**
関数に関する情報を得る（最大値，二乗平均，など）	**INFO_FONCTION**
（たとえば，XmGrace ソフトウェアのための）出力関数	**IMPR_FUNCTION**
場の量または結果からの関数の生成	**RECU_FONCTION**

- **例として zzzz100a のテストケースを参照のこと**

End of presentation

Is something missing or unclear in this document?
Or feeling happy to have read such a clear tutorial?

Please, we welcome any feedbacks about Code_Aster training materials.
Do not hesitate to share with us your comments on the Code_Aster forum dedicated thread.