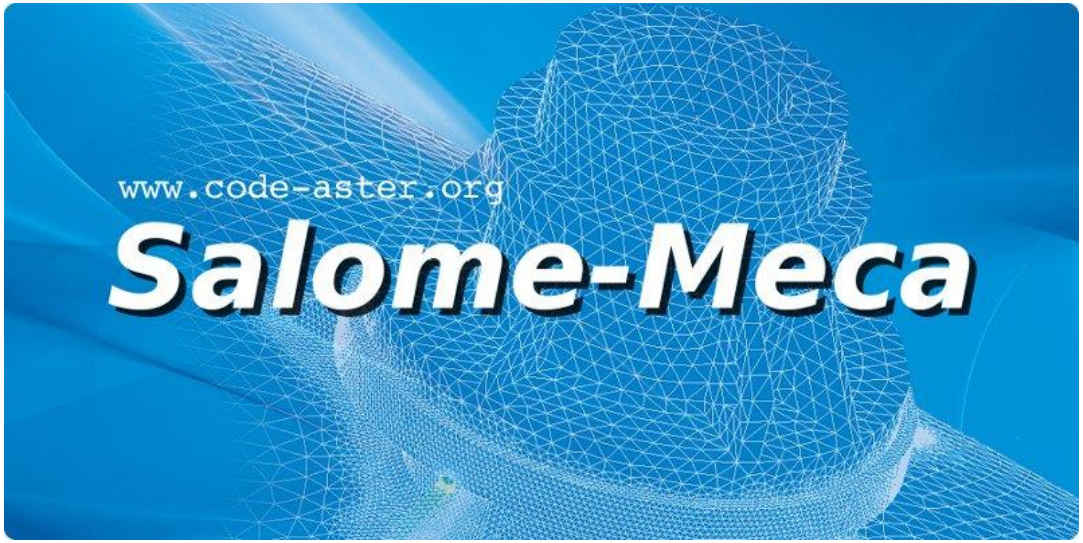
Functions and formulas



Code_Aster, Salome-Meca course material

Definitions

- **FONCTION** :

  tabulated (discrete) function depending on one parameter

- **NAPPE** :

  tabulated (discrete) function depending on two parameters

- **FORMULE** :

  continuum mathematic formula depending on several parameters

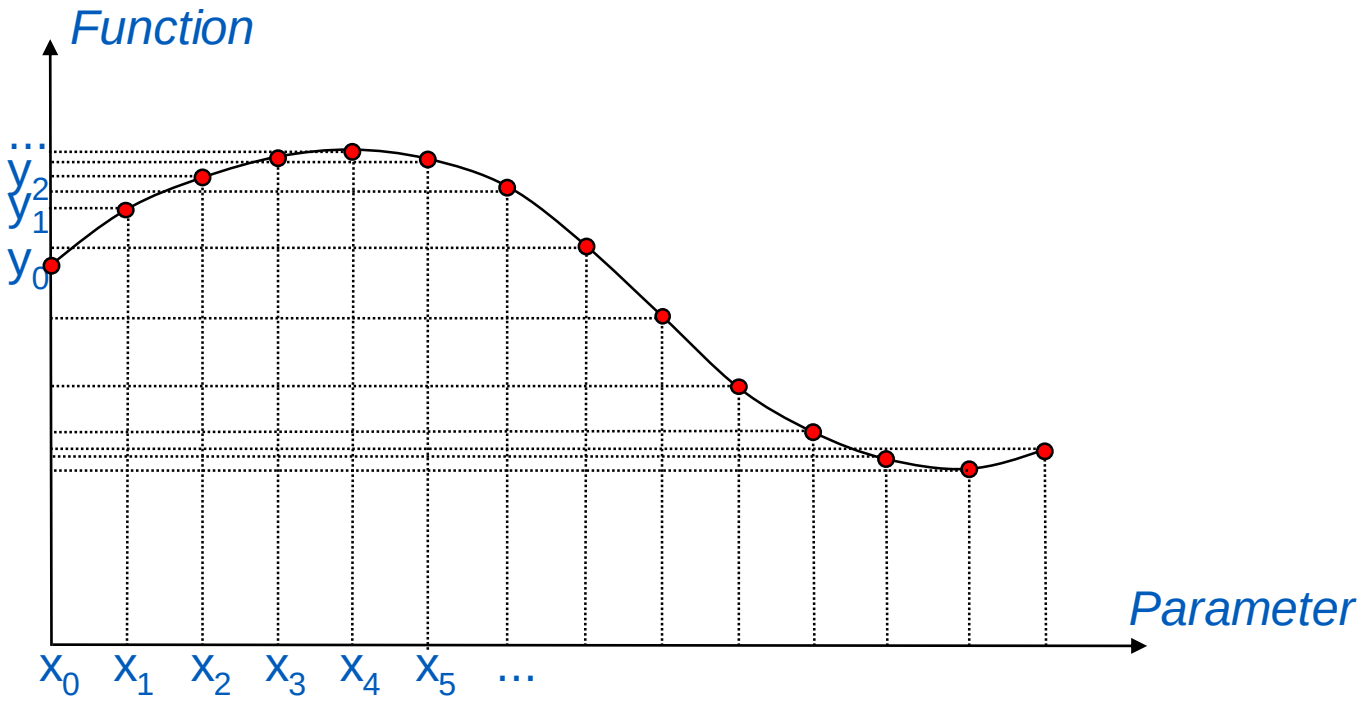
Parameters

- **Parameters : the most useful in *Code_Aster***

ABSC_CURV	Curvilinear abscissa	**EPSI**	Strain
DX	Displacement along X	**SIGM**	Stress
DY	Displacement along Y		
DZ	Displacement along Z		
DRX	Rotation around X	**INST**	Time
DRY	Rotation around Y		
DRZ	Rotation around Z		
X	Coordinate X	**TEMP**	Temperature
Y	Coordinate Y		
Z	Coordinate Z		

General principles of code and platform (1/4)

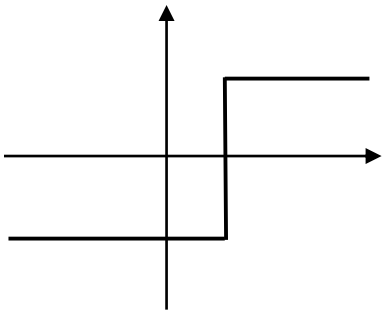
- **Function : definition by (x_i,y_i)**



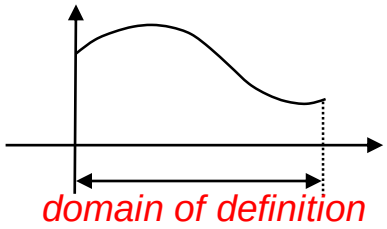
Abscissas values are stricly increasing : $x_0<x_1<x_2<x_3<x_4<x_5<x_6<x_7<x_8<...$

Function

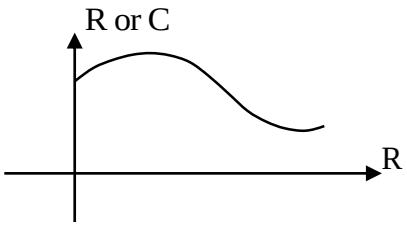
- **Function in *Code_Aster* : mathematical sense**



One abscissa value, several ordinate (y-axis) values:
→ NOT a function

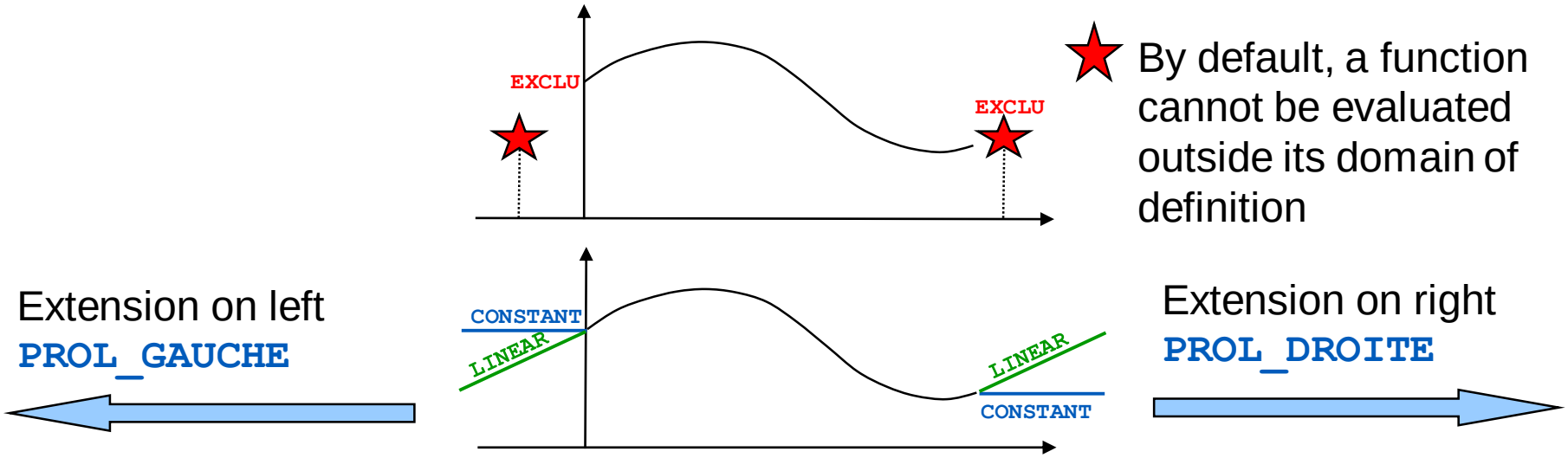


Function have a domain of definition



Function can be complex or real
Abscissa is **only real**

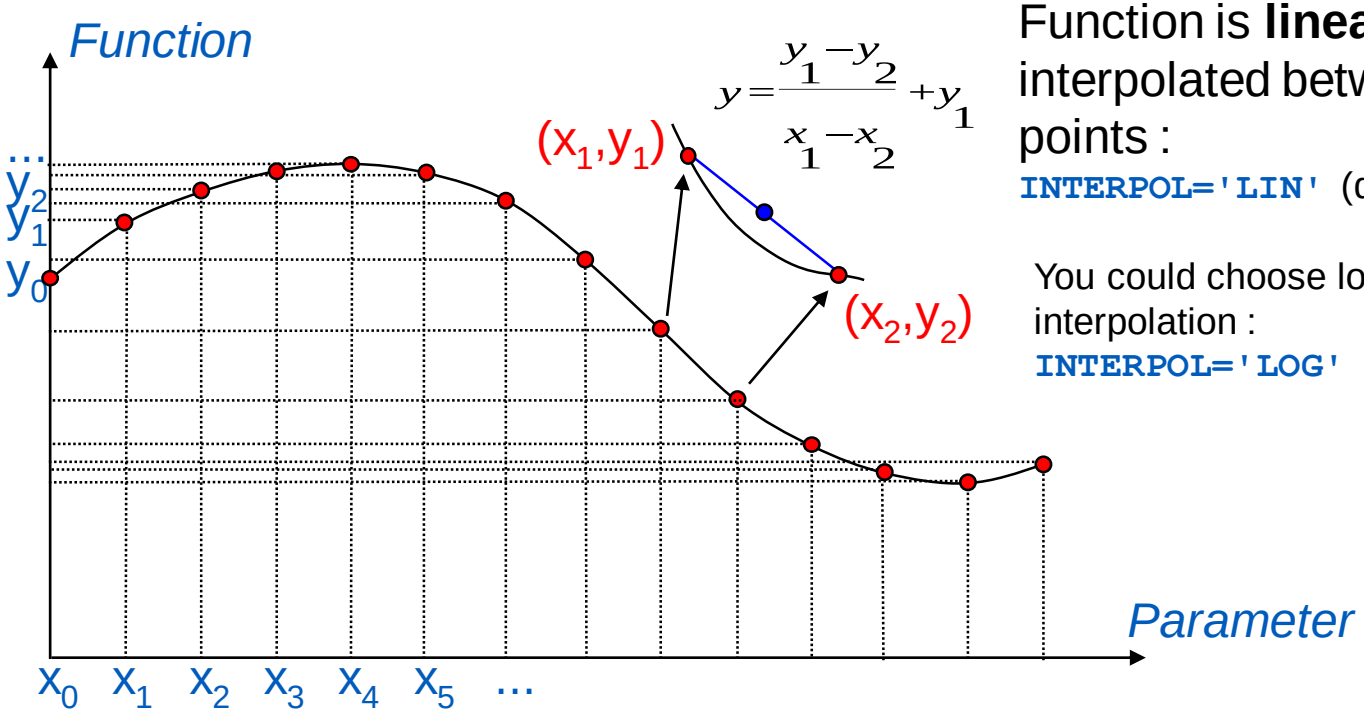
Function



EXCLU : no extension (default)
CONSTANT : extension by last value
LINEAIRE : extension by linear extrapolation

Function

Function : interpolation between points



Function is **lineary** interpolated between two points :

INTERPOL='LIN' (default)

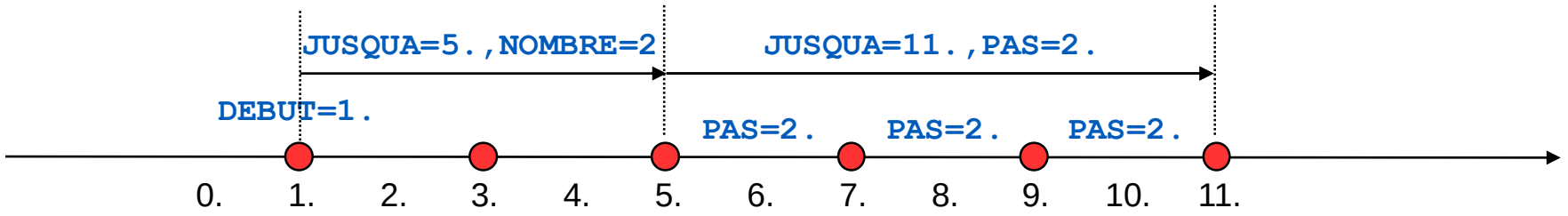
You could choose logarithmic interpolation :

INTERPOL='LOG'

List of real values

- **Defining list of real : DEFI_LIST_REEL**

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

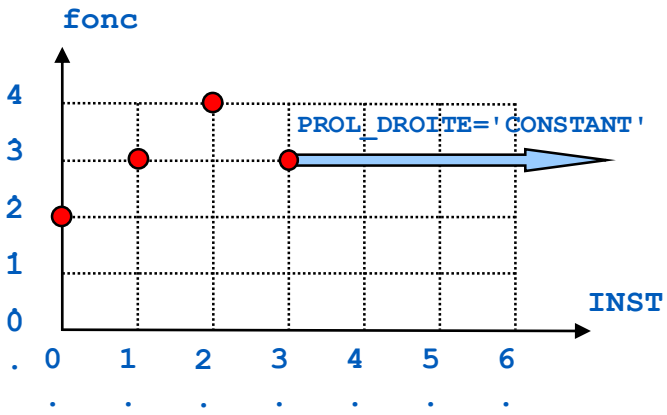



```
ListR = DEFI_LIST_REEL(VALE=(1.,3.,5.,7.,9.,11.))

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

Function

Function : Definition



Complex function :
VALE_C instead of **VALE**

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

```
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')
```

Abscissas increase

Function

- **Function with two parameters: DEFI_NAPPE**

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

Function of function : Previous defined functions

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

Function of function : Redefining functions in **DEFI_NAPPE**

- **See zzzz100a test-case for several examples**

Formula

- **Formula : defined by a mathematical function**

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

python™

« Using Python as a calculator »
http://docs.python.org/tut/tut.html
http://docs.python.org/lib/module-math.html

The main functions of Python math module are imported **by default** in *Code_Aster* (from math import * is not necessary)

Use triple quote `'''` to define formulas on several lines in Python

- **See zzzz100a test-case for several examples**

Formula

Formula is Python function

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

Python evaluation

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

Function of function

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

High-level function

```
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)''')
```

Function with several parameters

See zzzz100a test-case for several examples

Formula

- Transform **FORMULE** in **FONCTION : CALC_FONC_INTERP**

```
SI    = FORMULE(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')
```

- Transform **FORMULE** in **NAPPE : CALC_FONC_INTERP**

- See zzzz100a test-case for several examples

Remarks

- **Using `FONCTION` or `FORMULE` ?**
  - Function are tabulated : faster when using in low-level Fortran (behavior law for instance)
  - Formulas are « exact » : more precise
  - Defining `FORMULE` and use `CALC_FONC_INTERP` !

- **Where using functions ?**
  - Boundary conditions (`AFFE_CHAR_MECA_F` command for instance)
  - Behavior law : traction curve for elastoplastic, laws depending on temperature
  - Multiplicative functions for boundary conditions in non-linear

Other commands

Read function from file	**LIRE_FONCTION**
Get information about function (maximum, rms, etc.)	**INFO_FONCTION**
Print a function (for XmGrace software for instance)	**IMPR_FUNCTION**
Create function from results or from field	**RECU_FONCTION**

- **See zzzz100a test-case for several examples**

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.