

# CPEX0044: Encoding sets of functions in generic variables

Koen Hillewaert (koen.hillewaert@cenaero.be)

## 1 Introduction

CGNS currently can not encode data provided as analytical expressions nor functions. This functionality however would allow to

- Introduce interpolatory functions for high order meshes/solutions
- Specify expressions for boundary conditions
- Encode generic constitutive equations

Given the range of applications, one would like to

- specify independent variables, which can be predefined for a specific application (eg. for interpolation functions, we could expect the presence of parametric coordinates u,v,w).
- Specify parameter constants and macros for repeating expressions.
- Provide all functions supported by compilers

Typically not a single function is required, but rather coherent sets of functions should be defined.

Fortunately, there are a number of open source libraries that provide this functionality, using a syntax which is rather close to code notation in C or fortran. The proposal is to encode functions by using this syntax as it already constitutes a de facto standard and is at the same time fairly simple and easily readable.

## 2 Proposed extension of the SIDS

The extension introduces a new `FunctionSet_t` node, the children can be implemented using the `Descriptor_t` nodes. The `FunctionSet_t` are integrated in a dedicated node "FunctionSets" in `CGNSBase_t`.

### **2.1 FunctionSets leaf in CGNSBase\_t**

Name = Functions Type = DataType = I data = <number of function sets> Cardinality 0:1	
<b>Children</b>  name = Function sets datatype = FunctionSet_t dims = [N] data = <function sets> Cardinality = 1:1	<b>Comments</b>

## 2.2 FunctionSet\_t

The extension concerns the introduction of a new leaf type, FunctionSet\_t in the Functions\_t leaf in CGNSBase\_t. It is composed as:

name = <user defined> type = FunctionSet_t datatype = C1 data = <description of the function set> cardinality = 0:N	
Children	Comments
name = Variables type = Descriptor_t datatype = C1 dims = [N] data = <name of the variables> cardinality = 1:1	each line of data is a variable name
name = Functions type = Descriptor_t datatype = C1 dims = [N] data = <function expressions> cardinality = 1:1	each line of data is an expression
name = Parameters type = Descriptor_t dims = [N] data = <names of the parameters> cardinality 0:1	each line contains a parameter name, that can be used in the functions defined in “Functions”
name = ParameterValues type = Real_32 dims = [N] data = <values of the parameters> cardinality 0:1	each line contains a parameter value, in the order of the names in the “Parameters” block

Only one Variables, Parameters, ParameterValues and Functions blocks are allowed. Extra Descriptor\_t or UserDefined\_t nodes are allowed.

## 2.3 Supported expressions

The “Functions” block consists of a semicolon separated list of strings, each of which corresponds to a single function. The expressions can be any valid mathematical expression involving numbers, any of the variables, the parameters and the following ingredients:

- Structuring operators
  - Leading minus “-”
  - (Nested) bracketed expressions “(...)”
- Binary operators
  - Standard : “-”, “+”, “/”, “\*”
  - “%” (modulo)
  - “^” (power)
- Standard single argument functions
  - Exponential: “exp” (natural exponential), “log” (natural log), “log10” (base 10 log), “log2” (base 2 log)
  - Trigonometric: “sin”, “cos”, “tan”, “asin”, “acos”, “atan”, “arccsin”, “arccos”, “arctan”
  - Hyperbolic: “sinh”, “cosh”, “tanh”, “asinh”, “acosh”, “atanh”
  - Rounding: “round”, “floor”, “ceil”, “step”
- Predefined constants
  - Pi : “pi”, “Pi”
  - Natural exponent “e”

## 2.4 Example

The following Function\_t block describes the interpolation in a quadrilateral of 2nd order.

Interpolation_Quad_p2 [Function_t] [MT]		
	Variable [Descriptor_t][C1][*]	
		u v
Function [Descriptor_t][C1][*]		
	$(u-u^2)*(v-v^2)/4$ $-(u+u^2)*(v-v^2)/4$ $(u+u^2)*(v+v^2)/4$ $-(u-u^2)*(v+v^2)/4$ $(u^{2-1})*(v-v^2)/2$ $(u+u^2)*(v^{2-1})/2$ $(1-u^2)*(v-v^2)/2$ $(u-u^2)*(v^{2-1})/2$ $(u^{2-1})*(v^{2-1})$	

## 3 Interface in the mid-level library

### 3.1 Accessing function sets

The FunctionSet\_t is to be accessed using the general navigation functions, described in the section “Navigating a CGNS File”. Therefore, the functions need to allow a new keyword FunctionSet\_t. Eg. when we have several FunctionSet\_t in Base basenum, within file filenum. The third is named “InterpolationTet4”. E.g. to access the functions, one can use any of the following

```
cg_goto(filenum, basenum, "FunctionSets", 0, "FunctionSet_t", 1, "Functions", 0, NULL);
cg_goto(filenum, basenum, "FunctionSets", 0, "InterpolationTet4", 0, "Functions", 0, NULL);
cg_gopath(filenum, "/Base/FunctionSets/InterpolationTet4/Functions/");
```

Subsequently, to learn the number of functions one can use

```
int nbFunctions;
cg_ndescriptors(&nbFunctions);
```

And finally to get one of the functions, eg. the 5th

```
char* function5;
cg_descriptor_read(5, name, &function5);
```

### **3.2 Modifications to the manual**

#### **3.2.1 Manual node “Navigating a CGNS File”**

A specific task should be added to the following paragraph:

*“A few other nodes are not allowed to be deleted from the database because these are required nodes as defined by the SIDS, and deleting them would make the file non-CGNS compliant. These are:*

- ....
- *FunctionSet\_t: Function and Variable*