CPEX0045: Storing cell-wise polynomial data and generalising the description of curved grid elements

1 Motivation and scope

The aim is to cater for the large diversity of high order methods by including a description of the interpolation functions for both solution and mesh in the file, thereby only fixing the bare essentials: the coordinate system. This will:

- allow to use the native storage of the application, leading to furthermore to
  - A simpler, more robust and generic implementation of the drivers;
  - More efficient I/O by straightforwardly dumping data blocks;
  - Avoiding the loss of precision for very specific interpolation functions;
- avoid the need to redefine the format for each new interpolation type/order/…;
- cater for space-time methods as well as for ALE computations by including time in the functional expressions.

Currently it is assumed that high order interpolation is **restricted to unstructured mesh computations**, as structured meshes do not offer the possibility to individually list elements per order and moreover do not support curved elements. Both modal and nodal interpolations are supported.

2 Rationale

This proposal lifts the limitation of fixing the interpolation functions implicitly by imposing the position Lagrange control points as proposed for geometric interpolation/curved elements in CPEX0036 and CPEX0038. Instead the description of the functional space and its interpolant is integrated as metadata in the CGNS file in order to allow a high flexibility as to the choice of interpolants or even coordinates, an automatic procedure to allow for very high interpolation order and time-dependent interpolants.

3 Extension of the SIDS

3.1 Conventions

Modifications to the SIDS
● addition of a new section 3.4 High order interpolation.

In the remainder of this section, we introduce the different paragraphs (with numbering to be adapted) that should be added to the new section.

“The CGNS standard allows the user to specify their own interpolation approach for both elements and solution. The basic principles are

1. *The element coordinate system per element type is the only fixed convention;*
2. *For the solution interpolation per each element type and interpolation order, a separate interpolation block is added which provides one out of three choices*

   - *the set of control points for Lagrange interpolants*
   - *the ordered set of functions defined in parametric coordinates*
   - *the set of shape functions in Cartesian coordinates*

   The first two cover parametric interpolation, whereas the last covers modal/Cartesian interpolation
3. *The mesh is always defined using interpolations in parametric space, either by specifying the set of control points or the ordered list of parametric interpolants. The standard will only allow element types defined up to now in order not to modify the element connectivity description.*
4. *The interpolation is* ***not*** *supposed to be the same for the geometry as for the solution*
5. *In addition to the spatial coordinates, also time can be used as an independent variable*

The following sections describe how interpolations are specified. "

3.1.1 Interpolation type enum Interpolation_t

*"**InterpolationType_t** specifies how the high order interpolants for the solution are defined. InterpolationType_t can take four values:*

- ***ParametricLagrange** corresponds to a Lagrange interpolation, based upon a set of specified control point coordinates in parametric space for a standard interpolation space.*
- ***ParametricFunctions** corresponds to specifying explicitly the set of functions in parametric coordinates associated to expansion weights.*
- ***CartesianFunctions** corresponds to modal interpolation functions in a Cartesian coordinate system, centered on the element and parallel to the main axes. If no function set is defined, we fall back on the classical Pascal sets."*
- ***IsoParametric*** corresponds to using the same interpolation functions for the solution as for the mesh

3.1.2 High order parametric interpolation

***"Scope**: The parametric interpolation conventions can be used for specifying both curved elements, thereby superseding the standard conventions, as for the solution."*

3.1.2.1 Standard coordinate systems for parametric interpolation

"The parametric coordinate system is defined per element, and are defined following the figure 1.

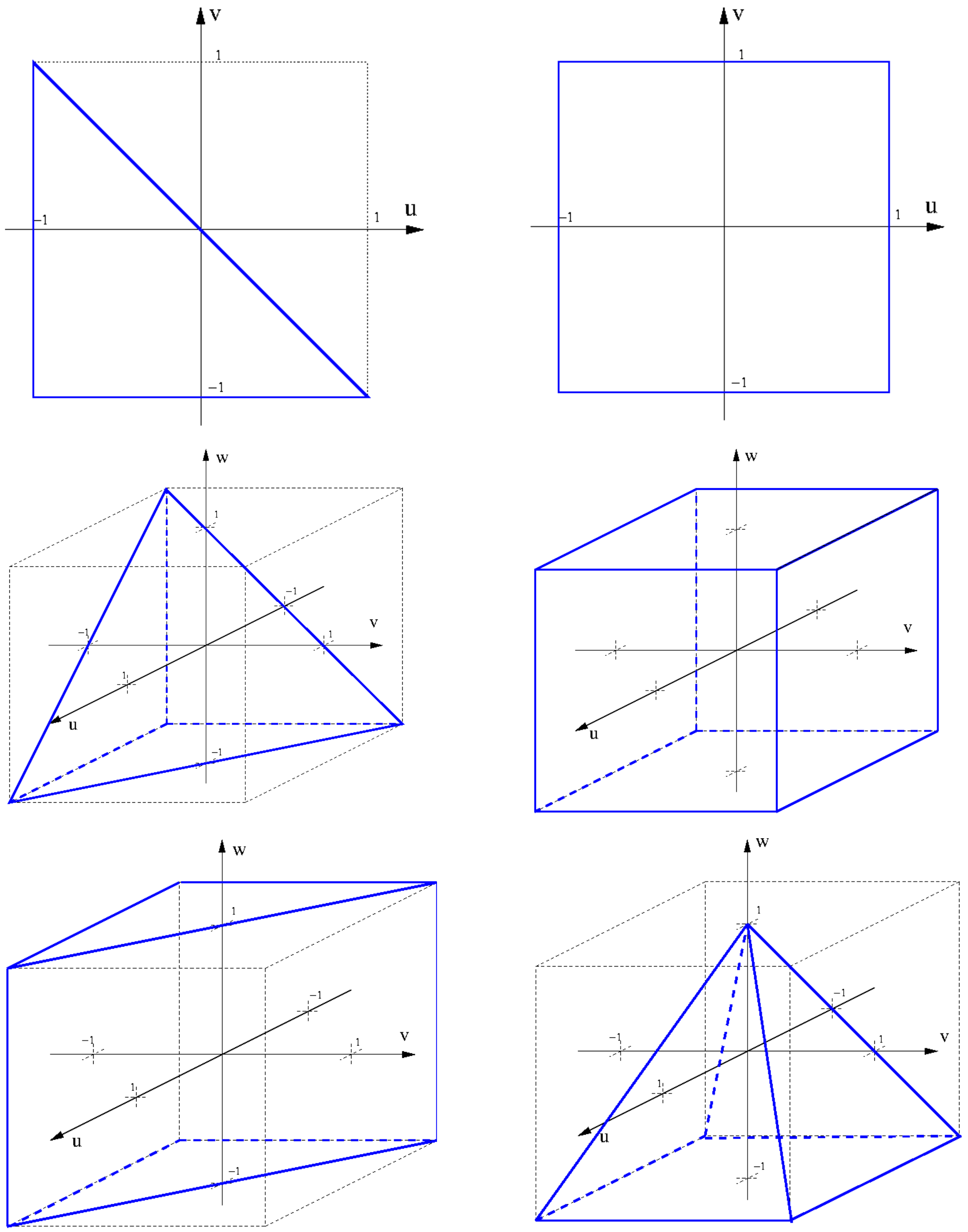

Figure 1: Parametric coordinate systems

For space-time computations, the coordinate system is extended with one dimension, by the tensor product of the spatial coordinate system, with the parameter interval [-1,1] in time. The latter corresponds to the physical time slab [t_n,t_{n+1}].”

3.1.2.2 Parametric interpolation by specification of Lagrange control points

*“**Scope**: Both solution and element shape can be specified using this formulation. For elements, this convention allows to redefine the position and order of the control points with respect to the*

standard definitions for each of the ElementType_t described in section 3.3. The standard definition will continue to be used in case no interpolation is explicitly introduced.

Lagrange interpolants require next to the specification of control point locations also the specification of the standard function space $\mathcal{V}$. Its cardinality N defines the number of control points that need to be specified.

We denote the Lagrange interpolant corresponding to control point $\mathbf{u}_i$ as λ_i[1]*. Furthermore, we need an arbitrary set of base functions for $\mathcal{V}$, such that $\mathcal{V}$= span(ψ_j, j=0..N-1). The Lagrange interpolants are then found as a linear combination of the base functions*

$$\lambda_i(\mathbf{u}) = \sum_j \mathbf{V}_{ij}^{-1}\,\psi_j(\mathbf{u})$$

Using the inverse of the Vandermonde matrix V associated to the control points u_i and the basis ψ

$$\mathbf{V}_{ij} = \psi_j(\mathbf{u}_i)$$

The (spatial) parametric function spaces $\mathcal{V}_p(u,v,w)$ for each element type and order p, which support the Lagrange type interpolation are listed in table 1. In Next to the standard "complete" element, also incomplete, or so-called serendipity elements are supported in higher dimensions. A first type of serendipity element only specifies control points on the edges. In three dimensional elements of sufficient order, a second serendipity interpolation can be defined which only excludes control points which are internal to the element. This classification is not univocal and in particular at lower order several elements can be classified in multiple classes. The element type tags are described in section 3.3 of the SIDS.

Element		*Function space $\mathcal{V}_p(u,v,w)$*		
Type	*Base type*	*Complete*	*Edge Serendipity*	*Face Serendipity*
Line	*BAR_2*	$\mathcal{L}_p(u)$ $N=p+1$	*n/a*	*n/a*
Quad	*QUAD_4*	$\mathcal{Q}^2_p(u,v)$ $N=(p+1)^2$	$(\mathcal{L}_p(u)\otimes\mathcal{L}_1(v) \oplus (\mathcal{L}_p(v)\otimes\mathcal{L}_1(u))$ $N=4p$	*n/a*
Hexa	*HEXA_8*	$\mathcal{Q}^3_p(u,v,w)$ $N=(p+1)^3$		
Triangle	*TRI_3*	$\mathcal{P}^2_p(u,v)$ $N=(p+1)(p+2)/2$		*n/a*
Tetra	*TETRA_4*	$\mathcal{P}^3_p(u,v,w)$ $N=(p+1)(p+2)(p+3)/6$		
Prism	*PENTA_6*	$\mathcal{P}^2_p(u,v)\otimes\mathcal{L}_p(w)$ $N=(p+1)^2(p+2)/2$		
Pyramid	*PYRA_5*	*See [BCD10]*		

[1] Although for 2D and 3D spaces multi-indices are more convenient, we will use for the simplicity of notation a single compounded index i=(i,j,k). Likewise a compound coordinate **u**=(u,v,w) is used.

Table 1 : List of Lagrange functional spaces per element and interpolation type. The base type and order - denoted p - are specified for solution interpolation, whereas the full element type is used to classify element interpolation.

In which we use direct sums $\oplus$ and products $\otimes$ of the following standard spaces of order p:

- *The linear space : $\mathcal{L}_p(u)$= span$\{u^i$, 0 <= i <= p}*
- *Tensor product spaces, N = $(p+1)^d$*
  - *$\mathcal{Q}^2{}_p(u,v) = \mathcal{L}_p(u)\otimes \mathcal{L}_p(v)$ = span$\{u^i\, v^j$, 0<=i+j<= p}*
  - *$\mathcal{Q}^3{}_p(u,v,w) = \mathcal{L}_p(u) \otimes \mathcal{L}_p(v) \otimes \mathcal{L}_p(w)$ = span $\{u^i\, v^j\, w^k$, 0<=i+j+k<= p}*
- *Pascal triangle/tetrahedron, N = (p+1) … (p+d)/d!*
  - *$\mathcal{P}^2{}_p(u,v)$ = span $\{u^i\, v^j$, 0<=i+j<= p}*
  - *$\mathcal{P}^3{}_p(u,v,w)$ = span $\{u^i\, v^j\, w^k$, 0<=i+j+k<= p}*

Note that in case the function space is defined in space-time (eg. for ALE meshes), the complete functional space is given by

$$\mathcal{V}_{p,q}(u,v,w,t) = \mathcal{V}_p(u,v,w) \otimes \mathcal{L}_q(u)$$

where p and q are the spatial and temporal order respectively."

3.1.2.3 Parametric modal interpolation

*"**Scope**: this type of interpolation only applies to solutions*

For solutions, also modal interpolation is allowed. The interpolation functions are to be described in a dedicated FunctionSet_t block, with variables u,v,w for space (in function of the dimension) and t for (parametric) time. The cardinality of the interpolation space is determined by the number of functions in the FunctionSet_t block.

***Example 1:** the Lagrange interpolants corresponding to the biquadratic quadrilateral are described by the following ordered set of functions*

```
(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);
```

***Example 2:** For Lagrange linear interpolation in 3D space and time[2] in the quadrilateral, we could have*

[2] The use of time as an independent variable would allow to store ALE meshes natively with full precision; however, this would imply storing different coordinate sets for each time step.

(1-u)(1-v)*(1-t)/8;*
(1+u)(1-v)*(1-t)/8;*
(1+u)(1+v)*(1-t)/8;*
(1-u)(1+v)*(1-t)/8;*
(1-u)(1-v)*(1+t)/8;*
(1+u)(1-v)*(1+t)/8;*
(1+u)(1+v)*(1+t)/8;*
(1-u)(1+v)*(1+t)/8;*
"

3.1.3 Cartesian modal interpolation

*"**Scope**: Cartesian modal interpolation only applies to solutions."*

3.1.3.1 Computation of the element coordinate system

"We specify a local Cartesian coordinate system per element, based upon the (simplified) barycenter. Say we note the global coordinates $R = X\ e_x + Y\ e_y + Z\ e_z$, we proceed by first computing the element barycenter R^e as the arithmetic mean of the locations of the principal vertices, ie. the nodes corresponding to those of the associated linear element:

$$\mathbf{R}^e = \frac{1}{N}\sum_{i=1}^{N}\mathbf{R}_i^e$$

The element local coordinates are then defined as $\mathbf{r} = \mathbf{R} - \mathbf{R}^e$; we then use the notation $\mathbf{r} = x\ e_x + y\ e_y + z\ e_z$.The variable names in the associated FunctionSet_t block should therefore contain x,y,z for space (in this order in function of the dimension) and t for (parametric) time."

3.1.3.2 Specification of modal interpolants

"The specification of the modal interpolants is done in a dedicated FunctionSet_t block. The variables should contain x,y and z for spatial coordinates and t for time. The cardinality of the interpolation is specified by the number of functions in the set.

***Example 1:** to describe to list the monomial interpolants corresponding to the biquadratic interpolation, the set of functions should be*

$1; x; y; x^2; xy; y^2; x^2y; xy^2; x^2y^2$

***Example 2:** For the quadratic Pascal space with linear time interpolants, we would have*

$1; x; y; x^2; y^2; xy; t; x\ t; x^2\ t; y^2\ t; x\ y\ t$

***Default**: In case no function set is defined we fall back on the (ordered) standard Pascal spaces*

- *1D: the monomials $1, x, x^2, x^3, \ldots.\ x^p$*
- *2D: The Pascal triangle ordered in the following way*

```
for (int i=0;i<=p;i++)
    for (int j=0;j<=i;j++)
        f[idx++] = $x^{(i-j)} y^j$
```

- *3D: the Pascal tetrahedron ordered following*

```
for (int i=0;i<=p;i++)
    for (int j=0;j<=i;j++)
        for (int k=0;k<=i-j;k++)
            f[idx++] = $x^{(i-j-k)} y^j z^k$
```
"

3.2 Overriding the element definition and solution interpolation

Modifications to the SIDS:
● include list of solution/element interpolants in section *12.6 Family_t* ● new section *12.10 ElementInterpolation_t* ● generalisation of section *7.3 Elements_t* and example in 7.4 ● new section *12.11 SolutionInterpolation_t* ● generalisation of section *7.7 FlowSolution_t* and example in 7.8 ● renumber sections 12.10 UserDefinedData_t and 12.11 Gravity_t

The following sections detail the modifications for each section separately:

3.2.1 modification of section 12.6

*"On a case by case basis, ie. per element type and interpolation order, one can provide alternative mesh and solution interpolants in a dedicated family to the zones in question. This in turn is implemented using dedicated list of **ElementInterpolation_t** and **SolutionInterpolation_t** leafs within **Family_t.***

```
Family_t :=
 {
   List(          Descriptor_t          Descriptor1          ...          DescriptorN          )          ;   (o)
   FamilyBC_t FamilyBC ;                                                                                     (o)
  ...
    List (ElementInterpolation_t Elementinterpolation1  ... ElementInterpolationN);                          (o)
    List (SolutionInterpolation_t SolutionInterpolation1 ...  SolutionInterpolationN);                       (o)
 } ;
```

ElementInterpolation_t and SolutionInterpolation_t are described in sections 12.10 and 12.11 respectively."

3.2.2 The description of ElementInterpolation_t, SIDS section 12.10

*"The **ElementInterpolation_t** specifies the geometric interpolation of an element, by listing an alternative set of Lagrange high order control points in parametric space following the element conventions for the coordinate system. In absence of such a block for a given ElementType_t, the*

standard following section 3.3 is followed. ***It is assumed that the first points correspond to the principal vertices of the corresponding linear element, in the same order.***

The ElementInterpolation_t leaf is defined as follows

```
ElementInterpolation_t :=
{
  ElementType_t Element;                                                        (r)
  DataArray_t<Float,DataSize[]>  LagrangePoints;                                (o)
};
```

***Limitations**: The current proposal maintains ElementType_t to describe both element type and geometric order, meaning we can not go beyond 4th order interpolation. This choice is motivated by maintaining the ElementConnectivity_t leaf in its current shape and the fact that currently there is no real need for higher geometric orders.”*

3.2.3 Changes in section 7.3 Elements_t

“In case an alternative location for the element control points are specified, the actual elements are defined as before by listing the indices in the coordinate table, with the notable change that the order will correspond to the control point coordinates or function set specified in the corresponding ElementInterpolation_t block. If a specific element type is not found amongst these leafs, the standard convention is supposed.”

3.2.4 Specification of the solution interpolants in new section 12.11 SolutionInterpolation_t

“The interpolation functions associated to the interpolation on a given element type and order are stored in a ***SolutionInterpolation_t*** *leaf, which is again attached to the corresponding Family.*

```
SolutionInterpolation_t :=
{
  ElementType_t Element;                                                        (r)
  Integer SpatialOrder;                                                         (r)
  Integer TemporalOrder;                                                      (o/d)
  Interpolation_t                                                               (r)
  DataArray_t<Float,DataSize[]>  LagrangePoints;                                (o)
  char*  FunctionSet;                                                           (o)
};
```

The relevant SolutionInterpolation_t block will be found using the pair composed by the (basic) element type and interpolation order. The former corresponds to either the actual element tag or, if the corresponding SolutionInterpolation_t block is absent, the type of the corresponding linear element. For instance, the interpolation functions for the 2nd order solution on a 4th order tetrahedron will be associated to element tag TETRA_35 or TETRA_4. Finally, if InterpolationName is not specified, standard interpolation (ie. constant per element) applies.”

3.2.5 Addition to section 7.7 FlowSolution_t

“In case variable high order solutions are stored, a separate zone per interpolation order in space (and time) should be stored. The interpolation orders attached to the zone are indicated by the

integers SpatialOrder resp. TemporalOrder. The location of the solution is then supposed to be CellCenter, and in case of a variable order solution, one needs to use PointRange or PointList to single out the elements which will use the specified order.

```
FlowSolution_t< int CellDimension, int IndexDimension,
                int VertexSize[IndexDimension],
                int CellSize[IndexDimension] > :=
{
  List( Descriptor_t Descriptor1 ... DescriptorN ) ;                          (o)
  GridLocation_t GridLocation ;                                               (o/d)
  int SpatialOrder;                                                           (o/d)
  int TemporalOrder;                                                          (o/d)
  Rind_t<IndexDimension> Rind ;                                               (o/d)
  IndexRange<IndexDimension> PointRange ;                                     (o)
  IndexArray<IndexDimension, ListLength[], int> PointList ;                   (o)
  List( DataArray_t<DataType, IndexDimension, DataSize[]>
      DataArray1 ... DataArrayN ) ;                                           (o)
  DataClass_t DataClass ;                                                     (o)
  DimensionalUnits_t DimensionalUnits ;                                       (o)
  List( UserDefinedData_t UserDefinedData1 ... UserDefinedDataN ) ;           (o)
} ;
```

The default value for SpatialOrder depends on the type of interpolation; the default for TemporalOrder is 0."

4 Extensions to the Mid-Level Library

4.1 Helper functions

Functions	Modes

ierr = cg_element_lagrange_interpolation_size(ElementType_t t)
ierr = cg_solution_lagrange_interpolation_size(ElementType_t t,int os,int ot)
	r - -
	r - -

Input/output parameters	
Parameter	Comments
et os	Element type Spatial interpolation order

These functions allow to get the cardinality of the Lagrange interpolation space for a given element type and potentially the interpolation order for both time and space.

4.2 Reading / encoding the interpolation characteristics the family interface.

Caveat: here two interfaces are proposed

- "Standard" interface: loop through interpolation spaces by index

Functions	*Modes*
ierr = cg_element_interpolation_read(int fn, int b,int fam,int en,ElementType_t t,double* pu,double* pv,double* pw)* *ierr = cg_nelement_interpolation_read(int fn,int bn,int fam,int en,int* ne)*	*r - -*
ierr = cg_element_interpolation_write(int fn,int b,int fam,ElementType_t,double pu,double*pv,double *pw)*	*- w m*
ierr = cg_solution_interpolation_type_read(int fn,int b,int fam,int sn,ElementType_t t,int* o, InterpolationType_t* it)* *ierr = cg_solution_interpolation_points_read(int fn,int b, int fam,int sn,double* pu,double* pv,double*pw,double* pt)* *ierr = cg_solution_interpolation_functions_write(int fn,int b,int fam,int sn,char* fset)*	*r - -* *r - -* *r - -*
ierr = cg_solution_interpolation_write(int fn,int b,int fam,ElementType_t t,int o,double pts)* *ierr = cg_solution_interpolation_write(int fn,int b,int fam,ElementType_t t,int o,char* setn)*	*- w m* *- w m*

- Alternative : find interpolation space as a function of element type and order

Functions	*Modes*
ierr = cg_element_interpolation_read(int fn, int b,int fam,int en,ElementType_t t,double pu,double* pv,double* pw)*	*r - -*
ierr = cg_element_interpolation_write(int fn,int b,int fam,ElementType_t,double pu,double*pv,double *pw)*	*- w m*
ierr = cg_solution_interpolation_type_read(int fn,int b,int fam,int sn,ElementType_t t,int os,int ot, InterpolationType_t it)* *ierr = cg_solution_interpolation_points_read(int fn,int b, int fam,ElementType_t et,int os,int ot,double* pu,double* pv,double*pw,double* pt)* *ierr = cg_solution_interpolation_functions_read(int fn,int b,int fam,ElementType_t,int os,char** fset)*	*r - -* *r - -* *r - -*
ierr = cg_solution_interpolation_write(int fn,int b,int fam,ElementType_t t,int os,int ot,double pts)* *ierr = cg_solution_interpolation_write(int fn,int b,int fam,ElementType_t t,int os,int ot,char** setn)*	*- w m* *- w m*

The parameters are the same for both interfaces

Input/output parameters	
Parameter	*Comments*
fn	*CGNS file index number*
bn	*base index number*
fam	*family index number*
ne	*number of element interpolation blocks*
en	*element interpolation index number*
ns	*number of solution interpolation blocks*
sn	*solution interpolation index number*
et	*element type*
os	*spatial interpolation order*
ot	*temporal interpolation order*
pu	*control points - u coordinate*

pv *pw* *pt* *fs*	*control points - v coordinate* *control points - w coordinate* *control points - t coordinate* *function set name*

And the accompanying text:

The family contains the set of interpolation bases:

- *for elements as a function of the element type ElementType_t*
- *for the solution in function of the combination ElementType_t and two interpolation orders os and ot. This means that no more than one specification can be present for the triplet (t,os,ot). The element type always refers back to the baseline element, ie. the solution interpolation basis for the pair (TETRA_4,4) and (TETRA_10,4) are the same*
- *the number of coordinates of the control points are defined by the dimension associated to the element type dimension*

4.3 Accessing data in the FlowSolution_t node

Functions	*modes*
cg_sol_interpolation_order_read(int fn, int bn, int zn,int sn,int order)* *cg_sol_interpolation_order_write(int fn,int bn,int zn,int sn,int order)*	*r - -* *- w m*

Input/output parameters	
Parameter	*Comments*
fn *bn* *zn* *sn* *o*	*CGNS file index number* *base index number* *zone index number* *solution block index number* *interpolation order*

"The interpolation order is assigned per solution block within an unstructured zone, whereas the details concerning the interpolation functions are encoded in the family attached to the zone. In this case, the solution is supposed to be attached to CellCenter; the values are in this case the expansion weights in the basis. The specific interpolation basis is defined through the combination of the element type and the interpolation orders.

The cardinality of the interpolation functions is to be determined first by accessing the description of the interpolation."

5 Extension to the SIDS file mapping

5.1 ElementInterpolation_t, child node of Family_t

Family_t

	Children		
	...		

	name = <user defined>
	label = **ElementInterpolation_t**
	datatype = I4
	data = <Element type>
	cardinality = 0:N		

		Children	**Comments**

		name = LagrangeControlPoints
		type = Descriptor_t
		datatype = R8
		dims = [2]
		data = <point locations>
		cardinality = 1:1
		parameters: Dimension, NumberOfPoints
			table [Dimension][NumberOfPoints]
			
			dimension corresponds to that of the element

5.2 SolutionInterpolation_t, child node of Family_t

Family_t			
	Children		
	...		

	name = <user defined>
	type = **SolutionInterpolation_t**
	datatype = I4
	dim=3
	data = <element type,spatial order, temporal order>
	cardinality = 0:N		

		Children	**Comments**

		name = InterpolationType
		datatype = **InterpolationType_t**
		data = <choice for interpolation type>
			LagrangeParametric, ModalParametric or ModalCartesian

		name = LagrangeControlPoints
		type = Descriptor_t
		datatype = R8
		dims = [2]
		data = <point locations>
			table [Dimension][NumberOfPoints]

		cardinality = 0:1 parameters: Dimension, NumberOfPoints	dimension corresponds to element and can be incremented by 1 for space-time

		name = FunctionSet
		type = Descriptor_t
		data = <name of the set of interpolation functions>
		cardinality = 0:1	

5.3 FlowSolution_t

A single child node will be added to FlowSolution_t

FlowSolution_t	
	Children
	....

	name = InterpolationOrders
	type = **IndexArray_t**
	datatype = I4
	dimensions = 1
	dimension values = 2
	data = <spatial and temporal interpolation order>
	cardinality = 0:1

6 References

[CPEX0038] CPEX0038 : “Quartic Elements for High Order”

[CPEX0036] CPEX0036 : “Cubic elements for High Order”

[CPEX00xx] CPEX00xx : “Encoding sets of functions in generic variables”

[BCD10] M.Bergot, G. Cohen and M. Duruflé, “Higher-order Finite Elements for Hybrid Meshes Using New Nodal Pyramidal Elements”, (2010), Journal of Scientific Computing 42, pp. 345–381, doi 10.1007/s10915-009-9334-9

[Mathex] The *mathex* library, written by S. Massago, is part of the *small scientific library (SSCILIB)* and can be found at http://sscilib.sourceforge.net