Publié dans Bulletin du Centre d'Hydrogéologie de
l'Université de Neuchâtel n° 11, pp. 113-124, 1992,
source qui doit être utilisée pour toute référence à ce travail

Technical note on the determination of local coordinates for given global coordinates in higher order finite elements

*par Laszlò Kiraly**

ABSTRACT

In many post-processing programs for Finite Element Models it is necessary to compute the local coordinates s^k corresponding to a given set of global coordinates x^i. For higher order finite elements the value of the local coordinates is obtained by solving a system of simultaneous non-linear equations, generally by using a computational scheme based on the Secant Method or the Newton Method.

The iterative technique described in this paper is equivalent to the Newton Method, but the operations are based on the geometric definition of the problem in 1-D, 2-D or 3-D curvilinear finite elements. The starting point is always located in the center of the parent element ($s^k = 0$), and the stepping toward the exact solution is explicitly carried out in the curvilinear local coordinate system. In moderately distorted elements the solution is generally obtained within 10 or 20 iterations and correct to within 10^{-10} global coordinate unit. The results are excellent even for 1-D or 2-D elements immerged in a 3-D global space. In this case the Jacobi matrix is not a square matrix and the standard library routines cannot cope with the problem.

A few numerical examples are presented to illustrate the procedure for 2-D and 3-D quadratic finite elements.

1. Definition of the problem

The reader is supposed to know the principle of the finite element method, as well as some basic concepts and notations commonly used in tensor analysis (see, for example KLINGBEIL 1966). To define the problem let us consider the highly deformed 8-node quadratic finite element in figure 1.

Each point of the element (for example, point P in figure 1) is simultaneously defined in a cartesian (orthonormal) global system by the global coordinates x^i, and in a curvilinear local system by the local coordinates s^k. Note that $-1 \leq s^k \leq +1$ or $0 \leq s^k \leq +1$. Each point s^k of the local space is mapped into a point x^i of the global space by the interpolation functions

* Centre d'hydrogéologie de l'Université de Neuchâtel; Rue E.-Argand 11, 2007 Neuchâtel (Suisse)

$N_n(s^k)$, which depend on the local coordinates only :

$$x^i(s^k) = N_n(s^k)\, X^{in} \qquad n=1,8; \quad i=1,2; \quad k=1,2 \qquad (1)$$

where X^{in} are the global coordinates of the nodal points. Throughout this paper we will use interpolation functions which are quadratic polynomials in the local coordinates s^k.

If it is easy to calculate the global coordinates x^i for a given set of local coordinates s^k, the converse is unfortunately not true. Let the point P in figure 1, for example, be given by its global coordinates. As the inverse of the interpolation functions $N_n(s^k)$ is not known

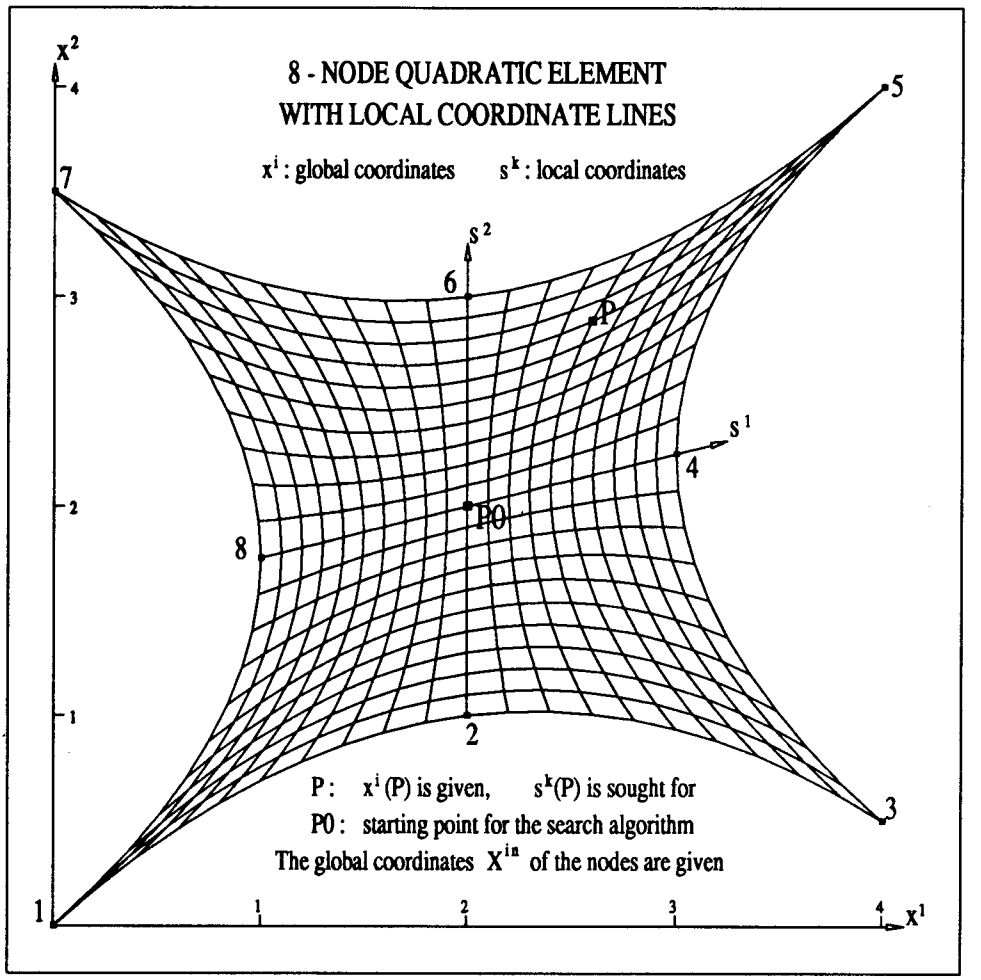

Figure 1 : Geometric representation of the problem

explicitly, the local coordinates of P can not be calculated directly, by using a simple formula like (1). The aim of this paper is to present a fairly general iterative procedure which makes it possible to find the local coordinates for a given set of global coordinates in higher order 1-D, 2-D and 3-D finite elements immerged in a 3-D global space.

2. Description of the iterative procedure

The local coordinate search algorithm is based on the geometric representation of the problem. The self-explanatory figure 2 represents an enlarged part of figure 1, where the point P is given by its global coordinates x_i (note that in a cartesian reference system $x_i = \delta_{ik}\, x^k$, i.e. the covariant and contravariant vector components are exactly the same). Let us define a starting point P0(so^k) in the center of the element. The local coordinates of P0 are known (so^k=0) and the corresponding global coordinates xo_i are easily calculated by the interpolation functions of (1):

$$so^k \quad \text{are known, and} \quad xo_i = X_{in}\, N^n(so^k) \qquad \text{at P0}$$

The difference vector $\vec{\Delta V}$, defined between P0 and P, is immediately given in the global system by

$$\vec{\Delta V} = \Delta x_i \vec{e}^{\,i} \tag{2}$$

where Δx_i=(x_i-xo_i) are the global components of $\vec{\Delta V}$ and $\vec{e}^{\,i}$ are the orthonormal base vectors. In the curvilinear local system the same vector is given by

$$\vec{\Delta V} = \Delta s^k \vec{a}_k \tag{3}$$

where $\vec{a}_k$ are the covariant base vectors at P0(so^k) and Δs^k represent the contravariant components of $\vec{\Delta V}$. Note that the covariant base vectors are tangent to the local coordinate lines everywhere (see figure 1 and figure 2), and their orientation and length vary from point to point in the interior of the element. The contravariant components Δs^k are obtained by the transformation of the known global components Δx_i:

$$\Delta s^k = \Delta x_i \frac{\partial x^i}{\partial s^j} g^{jk} \tag{4}$$

where [$\partial x^i / \partial s^j$] is the Jacobi transformation matrix and g^{jk} is the contravariant metric tensor, both calculated at the starting point P0(so^k). The relation (4) is exact at the starting point and holds approximately in its immediate vicinity. The next section contains additional information on how to calculate Δs^k explicitely.

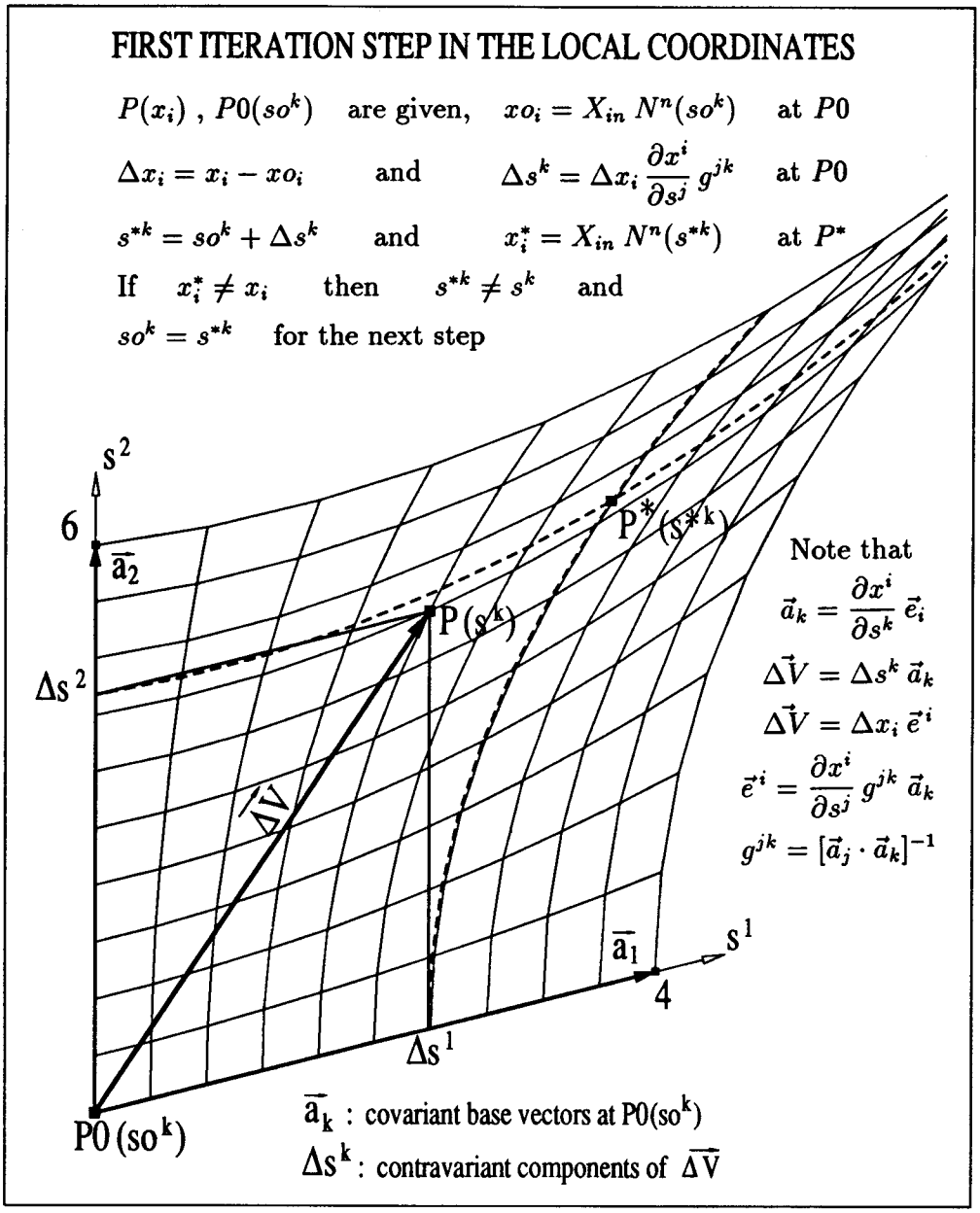

Figure 2 : Geometric representation of the search algorithm

It is now possible to make a "step" in the local coordinates simply by adding the components Δs^k to the local coordinates so^k of the starting point. We obtain a new point P* with the

following local and global coordinates (see figure 2) :

$$s^{*k} = so^k + \Delta s^k \qquad \text{and} \qquad x^*_i = X_{in} N^n(s^{*k}) \tag{5}$$

Obviously, if x^*_i are equal to the given global coordinates x_i, then s^{*k} are equal to the local coordinates s^k sought for. If not, then point P* becomes the new starting point for the next step :

$$so^k = s^{*k} \qquad \text{and} \qquad xo_i = x^*_i \qquad \text{for the next step}$$

The iterative procedure is summarized in figure 2 and in the flow chart of figure 3. The stepping toward the point P will be set forth until the difference $|x^*_i - x_i|$ becomes less than a predefined tolerance value (generally TOLER is about 10^{-14} to 10^{-12} global coordinate unit). Thanks to formula (4), derived from the geometric representation of the problem, the search algorithm described above is quite general and it is valid for all elements, provided the dimension of the local space (i.e. the dimension of the element) is less than, or equal to the dimension of the global space. In other words, the method may be applied even for 1-D and 2-D elements immerged in a 3-D global space, and associated with 3-D elements.

It is possible to *improve the search algorithm* considerably by diminishing the step length Δs^k in highly deformed elements, with strongly curved local coordinate lines. In this case the user may define a LIMIT value for the maximum step length, in most cases between 0.1 and 0.5 local coordinate units. This kind of "careful approach" gives excellent results even if the point $P(x_i)$ is located in the vicinity of a singularity, where the coordinate lines become parallel to each other and the Jacobian $\det [\partial x^i / \partial s^k] = 0$ (see for example the nodal points 1 and 5 in figure 1 and the obtained results in table 1A). The correction of the steplength is defined simply by the following program section :

```
LIMIT = 0.5  (user given value)
CORRECT = 1.0
IF (MAX(Δs^k) > LIMIT)  then  CORRECT = LIMIT / MAX(Δs^k)
Δs^k = CORRECT * Δs^k
```

The value of LIMIT should never exceed 1.0 local coordinate unit. The identification of the element in which the point $P(x_i)$ is located goes hand in hand with the local coordinate search algorithm. If the calculated local coordinates $s^{*k} \approx s^k$ are inside the interval [-1, +1], the point $P(x_i)$ belongs to the element under examination and the algorithm is stopped. If this is not the case, the local coordinates will be searched for in the neighbouring element.

As the algorithm is based on the geometric representation of the problem, each user may easily introduce improvements or "useful tricks" in the code, depending on the type of elements used in his model. Singularities (i.e. points where $\det [\partial x^i / \partial s^k] = 0$) should never occur in an element, for example. But if they do, and if one of them should coincide with one of the $P0(so^k)$, then the automatic re-location of the starting point by a very small amount is a

"useful trick" which makes it possible to continue the algorithm and eventually to obtain the desired result.

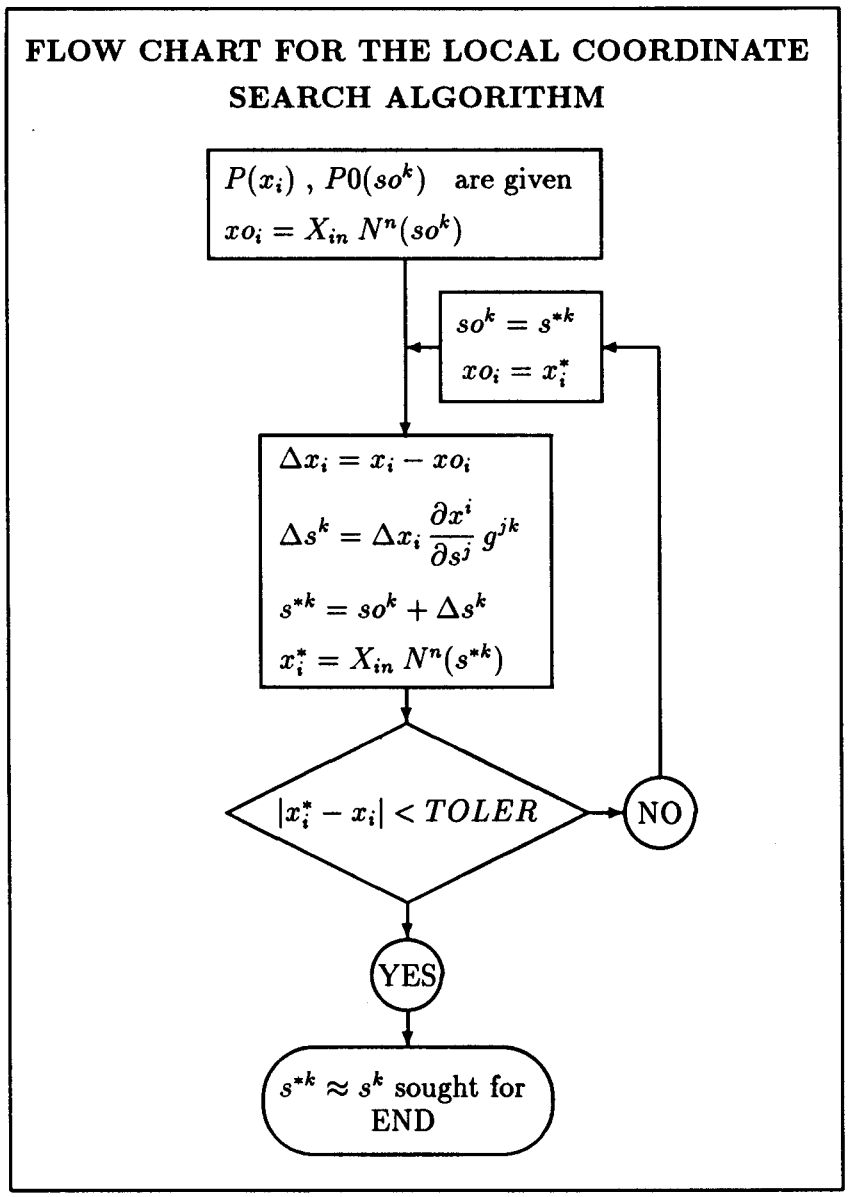

Figure 3 : Flow chart for the search algorithm

3. Remarks on the determination of the components Δs^k

As a matter of fact, the iterative procedure above described is equivalent to the Newton method if the dimension of the local space is the same as the dimension of the global space. Indeed, the problem may be written as

$$f^i(s^k) = \left[x^i - X^{in} N_n(s^k)\right] = 0$$

This is a set of 2 or 3 non-linear equations (in 2-D or 3-D global space) which has to be solved for s^k, all variables beeing the same as before. An iteration step of the Newton method is defined by

$$s^{*k} = so^k - \left[\frac{\partial f^i}{\partial s^k}\right]^{-1} \Delta x^i \qquad \text{with} \qquad \frac{\partial f^i}{\partial s^k} = -\frac{\partial N_n}{\partial s^k} X^{in} = -\frac{\partial x^i}{\partial s^k}$$

Finally we have

$$s^{*k} = so^k + \left[\frac{\partial x^i}{\partial s^k}\right]^{-1} \Delta x^i = so^k + \Delta s^k \qquad \text{and} \qquad \Delta s^k = \left[\frac{\partial x^i}{\partial s^k}\right]^{-1} \Delta x^i \qquad (6)$$

Both formula (4) and formula (6) make it possile to calculate the contravariant components Δs^k, but formula (6) is more restrictive than (4). It works only if the number of the independent local coordinates is the same as the number of the cartesian global coordinates, that is if we have only 3-D elements in a 3-D global space or only 2-D elements in a 2-D global space.

When 1-D or 2-D elements are mapped into a 3-D global space, the number of the independent local coordinates will be less than the number of the global coordinates, the Jacobi matrix $[\partial x^i / \partial s^k]$ will not be an invertible square matrix and formula (6) cannot be used to calculate the contravariant components Δs^k. Obviously, the Newton method could not be used to solve the problem, either.

To make possible the association of 1-D, 2-D and 3-D elements in the models, the direct inversion of $[\partial x^i / \partial s^k]$ should be avoided in most cases. As we have shown in earlier papers, (KIRALY 1979, 1985, 1988), and as can be seen in formula (4), this is possible provided we make use of a few elementary transformation laws between the cartesian global system and the curvilinear local reference system.

There are two ways to obtain the contravariant components Δs^k in the local system :

- First, we may express the global base vectors $\vec{e}^i$ as a function of the covariant local base vectors $\vec{a}_k$ and replace this expression into equation (2).
- Second, we may express Δs^k as a function of the global components Δx_i and replace it into

equation (3).

To do this, let us recall two standard transformation laws which do not contain the inverse of the matrix $[\partial x^i / \partial s^k]$ (see for example KLINGBEIL 1966) :

$$\vec{e}^{\,i} = \frac{\partial x^i}{\partial s^k} \vec{a}^{\,k} \qquad (7) \qquad \text{and} \qquad \Delta s_k = \frac{\partial x^i}{\partial s^k} \Delta x_i \qquad (8)$$

They are not exactly what we need : in (7) we have the contravariant base vectors $\vec{a}^k$ instead of the covariant base vectors $\vec{a}_k$, and in (8) we have the covariant components Δs_k instead of the contravariant components Δs^k. Fortunately, covariant components and base vectors may be transformed into contravariant components and base vectors when multiplied by the contravariant metric tensor g^{jk}. Thus $\vec{a}^k = g^{jk} \vec{a}_j$ will be replaced in (7), and both sides of (8) will be multiplied by g^{jk} to obtain $\Delta s^k = g^{jk} \Delta s_j$:

$$\vec{e}^{\,i} = \frac{\partial x^i}{\partial s^j} g^{jk} \vec{a}_k \qquad (7a) \qquad \text{and} \qquad \Delta s^k = g^{jk} \frac{\partial x^i}{\partial s^j} \Delta x_i \qquad (8a)$$

Equation (8a) gives directly the contravariant components Δs^k sought, even if $[\partial x^i / \partial s^j]$ is not a square matrix. We obtain exactly the same result if (7a) is replaced in equation (2) :

$$\overrightarrow{\Delta V} = \Delta x_i \vec{e}^{\,i} = \Delta x_i \frac{\partial x^i}{\partial s^j} g^{jk} \vec{a}_k = \Delta s^k \vec{a}_k \qquad \text{with} \qquad \Delta s^k = \Delta x_i \frac{\partial x^i}{\partial s^j} g^{jk} \qquad (8b)$$

In both cases we have to determine the contravariant metric tensor g^{ik}, but it can be obtained easily from the matrix $[\partial x^i / \partial s^k]$ itself :

$$g^{ik} = [g_{ik}]^{-1} = [\vec{a}_i \cdot \vec{a}_k]^{-1} \qquad \text{with} \qquad \vec{a}_k = \vec{e}_i \frac{\partial x^i}{\partial s^k}$$

In fact, the columns of the matrix $[\partial x^i / \partial s^k]$ represent the global components of the covariant base vectors $\vec{a}_k$. The matrix g_{ik} is the covariant metric tensor, a symetric and invertible square matrix. In practice, the calculations are much simpler than they seem at first sight. To make the operations more transparent let us apply equation (8a) to an 8-node 2-D element immerged in a 3-D global space, and let us introduce a more familiar notation for the variables : $s=s^1$, $t=s^2$, $x=x^1$, $y=x^2$, $z=x^3$ and $X_1 \ldots X_8$, $Y_1 \ldots Y_8$, $Z_1 \ldots Z_8$ will be the global coordinates of the nodal points.

In this particular case the difference vector $\overrightarrow{\Delta V}$ has only two contravariant components $[\Delta s \ \Delta t]$ in the local space, whereas it is defined by three global components $[\Delta x \ \Delta y \ \Delta z]$ in the cartesian global space. Equation (9) shows explicitly the very simple matrix operations necessary to transform the global components $[\Delta x \ \Delta y \ \Delta z]$ into the local components $[\Delta s \ \Delta t]$ (note the explicite form of $[\partial x^i / \partial s^j]$: it is 2x3 matrix).

$$\begin{bmatrix}\Delta s\\ \Delta t\end{bmatrix}=\begin{bmatrix} g_{11} & g_{12}\\ g_{21} & g_{22}\end{bmatrix}^{-1}\begin{bmatrix}\frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} & \frac{\partial z}{\partial s}\\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t}\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta y\\ \Delta z\end{bmatrix} \qquad (9)$$

with

$$g_{11}=\left(\frac{\partial x}{\partial s}\right)^2+\left(\frac{\partial y}{\partial s}\right)^2+\left(\frac{\partial z}{\partial s}\right)^2$$

$$g_{22}=\left(\frac{\partial x}{\partial t}\right)^2+\left(\frac{\partial y}{\partial t}\right)^2+\left(\frac{\partial z}{\partial t}\right)^2$$

$$g_{12}=g_{21}=\frac{\partial x}{\partial s}\frac{\partial x}{\partial t}+\frac{\partial y}{\partial s}\frac{\partial y}{\partial t}+\frac{\partial z}{\partial s}\frac{\partial z}{\partial t}$$

and

$$\begin{bmatrix}\frac{\partial x}{\partial s} & \frac{\partial x}{\partial t}\\ \frac{\partial y}{\partial s} & \frac{\partial y}{\partial t}\\ \frac{\partial z}{\partial s} & \frac{\partial z}{\partial t}\end{bmatrix}=\begin{bmatrix} X_1 \dots X_8\\ Y_1 \dots Y_8\\ Z_1 \dots Z_8\end{bmatrix}\begin{bmatrix}\frac{\partial N_1}{\partial s} & \frac{\partial N_1}{\partial t}\\ \vdots & \vdots\\ \vdots & \vdots\\ \frac{\partial N_8}{\partial s} & \frac{\partial N_8}{\partial t}\end{bmatrix}$$

The only "extra work" which we have to do with respect to the standard method is the calculation of the metric tensors g_{ik} and g^{ik}, but this extra work is largely compensated by the generality thus gained. This technique could be applied quite generally in the finite element method each time we need the inverse of the Jacobi matrix. It must be emphasized again, that sticking to the direct inversion of $[\partial x^i / \partial s^j]$ hinders many modellers from developing more adequate groundwater flow models for fractured and karstic aquifers with complicated geological structures, where the association of 1-D, 2-D and 3-D elements would be necessary to solve the problem. That being said, in the "normal case" (i.e. 2-D elements in a 2-D global space or 3-D elements in a 3-D global space) equations (4), (6), (8a) and (8b) give exactly the same results.

4. Numerical examples

The search algorithm was applied to the elements represented in figure 4 and the results are listed in table 1. Note that x, y, z are the given global coordinates and s, t, u are the local coordinates sought for.

Element (a) in figure 4 is exactly the same as the element represented in figure 1 and figure 2. The first line in table 1A gives the local coordinates for point P in figure 1 : s=0.4 and t=0.7 were obtained in only 5 iterations correct to within less than 10^{-15} global coordinate unit. Lines 2 and 3 represent the results for the nodal points 1 and 5 in figure 1. The exact local coordinates were obtained in hardly more than 20 iterations, in spite of the fact that we have two singularities (that is, det $[\partial x^i / \partial s^k] = 0$) at these nodal points. Line 6 gives the local coordinates for a point which is located outside of the element (s<-1.0).

The nodal points of the 2-D element (b) in figure 4 are not co-planar and the element is actually a curved surface in a 3-D space. If a point given by its global coordinates (x,y,z) is located exactly on that surface, then there is an exact solution to the problem and the corresponding local coordinates are obtained in a few iterations (see table 1B). If the target point P(x,y,z) is located outside (below or above) the curved surface, then there is no exact

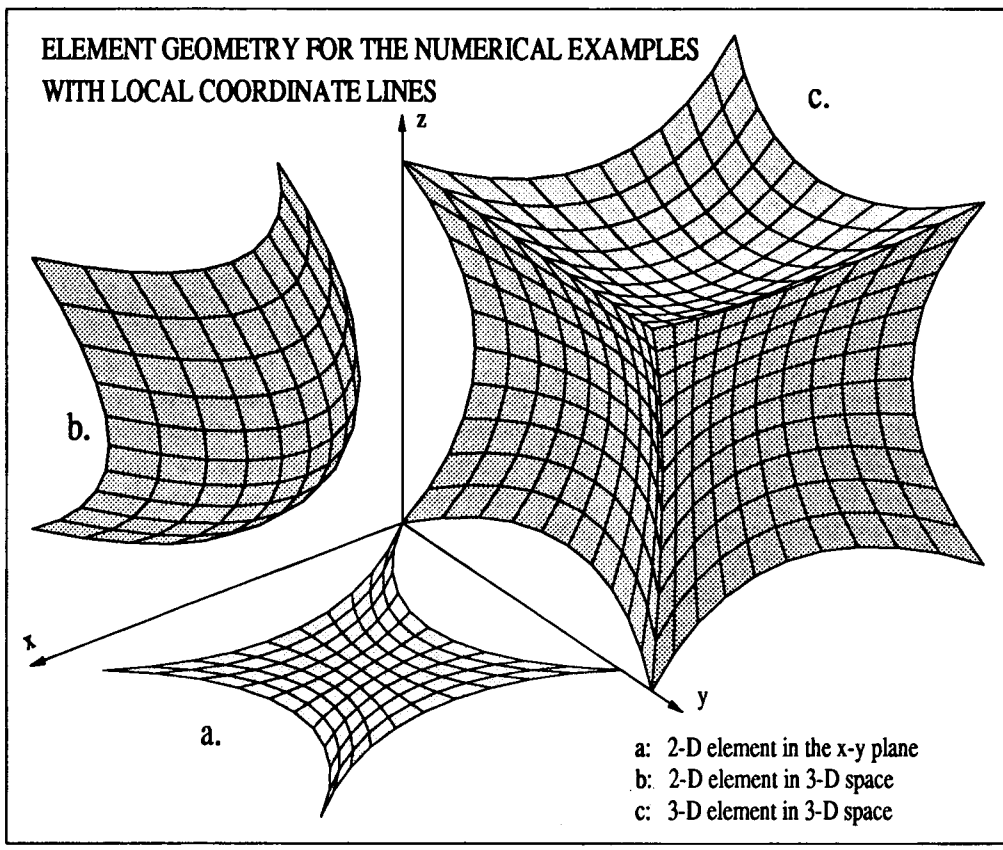

Figure 4 : Elements used for the numerical examples

TABLE 1A: ELEMENT (a) IN FIGURE 4.

N	x	y	iter	s	t	error
1	2.596	2.884	5	0.40000	0.70000	0.167E-15
2	0.000	0.000	21	-1.00000	-1.00000	0.354E-14
3	4.000	4.000	23	1.00000	1.00000	0.261E-14
4	4.000	0.500	2	1.00000	-1.00000	0.278E-16
5	0.000	3.500	2	-1.00000	1.00000	0.278E-16
6	0.500	2.000	6	-1.47108	0.14021	0.555E-16
7	1.500	2.000	4	-0.49460	0.10448	0.222E-15
8	2.857	1.239	6	0.55925	-0.72966	0.555E-16
9	1.239	2.857	6	-0.44482	0.84309	0.555E-16

TABLE 1B: ELEMENT (b) IN FIGURE 4

N	x	y	z	iter	s	t	error
1	3.000	2.500	3.000	0	0.00000	0.00000	0.000E+00
2	1.500	0.000	1.500	2	-1.00000	-1.00000	0.000E+00
3	1.500	0.000	4.500	2	-1.00000	1.00000	0.000E+00
4	4.500	0.000	4.500	2	1.00000	1.00000	0.000E+00
5	2.625	1.561	1.755	6	-0.25000	-0.83000	0.555E-16
6	4.313	1.404	3.499	16	0.87500	0.33300	0.212E-06
	No improvement during the last 10 iterations!						
7	2.025	1.719	3.675	5	-0.65000	0.45000	0.000E+00
8	3.600	1.688	4.050	5	0.40000	0.70000	0.222E-15
9	3.150	1.475	4.350	6	0.10000	0.90000	0.111E-15

TABLE 1C: ELEMENT (c) IN FIGURE 4.

N	x	y	z	iter	s	t	u	error
1	4.000	4.000	0.000	14	-5.93140	0.10755	-0.10755	0.888E-15
2	-4.000	4.000	0.000	2	1.00000	1.00000	-1.00000	0.000E+00
3	0.000	0.000	0.000	2	-1.00000	-1.00000	-1.00000	0.000E+00
4	-2.000	2.000	2.000	0	0.00000	0.00000	0.00000	0.000E+00
5	-2.234	1.956	2.435	4	0.21447	-0.03952	0.42490	0.277E-16
6	-3.256	1.523	0.876	6	0.94945	-0.27479	-0.75513	0.555E-16
7	-5.200	1.500	3.200	9	3.12379	-0.08475	0.20399	0.222E-15
8	-1.289	2.567	3.432	6	-0.38103	0.29941	1.28153	0.555E-16
9	-1.289	2.567	2.951	6	-0.51266	0.39351	0.78671	0.555E-16

Table 1 : Numerical results obtained by the search algorithm applied to the elements of figure 4. Note that X, Y, Z are the given global coordinates and s, t, u are the local coordinates sought for.

solution to the problem, because it is impossible to find 2 local coordinates corresponding exactly to this point. Instead of an exact solution, the search algorithm yields the best approximate solution $P^*(s^*,t^*)$ or $P^*(x^*,y^*,z^*)$. The point P^* is still on the element surface, but it is the nearest point of the element surface with respect to the given point P(x,y,z). In other words, P is located on the normal to the element surface at $P^*(s^*,t^*)$. Obviously, the distance between P and P^* will never be less than TOLER, and the algorithm is stopped when this distance is not improved during the last 10 iterations (see line 6 in table 1B). As a matter of fact, the best approximate solution is found when the components [Δs Δt] are zero in the local space.

The 3-D element (c) in figure 4 is not excessively deformed and the local coordinates are obtained in less than 20 iterations, even if the given point is well outside the element (see lines 1, 7 and 8 in table 1C).

The examples were run on a VAX780 computer and the calculation of the local coordinates took about 0.01 to 0.03 CPU seconds for a point, depending on the dimension of the element and the global space.

5. Conclusion

The search algorithm presented in this paper is an efficient and quite general iterative procedure which makes it possible to find the local coordinates for a given set of global coordinates in higher order 1-D, 2-D and 3-D finite elements immerged in a 3-D global space. It is based on the geometric representation of the problem and it avoids the complications related to the direct inversion of the transformation matrix $[\partial x^i / \partial s^k]$. This technique could be applied quite generally in the finite element method each time we need the inverse of the transformation matrix. We have to emphasize that sticking to the direct inversion of $[\partial x^i / \partial s^k]$ hinders many modellers from developing more adequate groundwater flow models for fractured and karstic aquifers with complicated geologic structures, where the association of 1-D, 2-D and 3-D elements would be necessary to solve the problem.

References

KIRALY L. 1979 : Remarques sur la simulation des failles et du réseau karstiques par éléments finis dans les modèles d'écoulement. – *Bulletin du Centre d'hydrogéologie de l'Université de Neuchâtel, No 3-1979, 155-167.*

KIRALY L. 1985 : FEM301 - A three dimensional model for groundwater flow simulation. –*NAGRA NTB 84-49, 96 p. + 40 p. listing.*

KIRALY L. 1988 : Large scale 3-D groundwater flow modelling in highly heterogeneous geologic medium. – *In : "Groundwater Flow and Quality Modelling" (editors : CUSTODIO, E. et alii.), 761-775, Reidel Publishing Company.*

KLINGBEIL E. 1966 : Tensorrechnung für Ingenieure. *Bibl.Inst. Mannheim, 231 p.*