

PRE-PUBLICACIONES DEL SEMINARIO MATEMATICO

2006

"GARCÍA DE GALDEANO"

A simple Matlab implemen-
tation of the Argyris
element

Víctor Domínguez
Francisco-Javier Sayas

N. 25



seminario
matemático

garcía de galdeano

Universidad de Zaragoza

A simple Matlab implementation of the Argyris element

Víctor Domínguez⁽¹⁾ Francisco-Javier Sayas⁽²⁾

⁽¹⁾Dep. Matemática e Informática, Universidad Pública de Navarra,
Campus de Arrosadía s/n, Pamplona 31006, Spain.

Email: `victor.dominguez@unavarra.es`

⁽²⁾Dep. Matemática Aplicada, Universidad de Zaragoza,
C.P.S., C./ María de Luna, no 3, Zaragoza 50018, Spain.

Email: `jsayas@unizar.es`

September 2006

Abstract

In this work we propose a new algorithm to evaluate the basis functions of the Argyris finite element and their derivatives. The main novelty here is an efficient way to calculate the matrix which gives the change of coordinates between the bases of the Argyris element for the reference and for an arbitrary triangle. This matrix is factored as the product of two rectangular matrices with a strong block structure which makes their computation very easy. We show and comment an implementation of this algorithm in Matlab. This implementation is validated in the last section, where two numerical experiments, an interpolation on a triangle and the finite element solution of the Dirichlet problem for the bilaplacian, are presented.

Category and Subject Descriptors: G.4 [**Mathematical software**] – *documentation*; G.1.8 [**Numerical Analysis**] Partial Differential Equations – *Finite element methods*.

General terms: Algorithms, Documentation.

Additional Key Words and phrases: Argyris element, finite elements, Matlab.

1 Introduction

The Argyris element ([1, 5, 6, 7]) is possibly the best-known and most widely used of all finite elements of class \mathcal{C}^1 . Finite elements of class \mathcal{C}^1 are commonly employed for conforming approximation of primal formulations of linear Partial Differential Equations of order four, such as the Kirchhoff plate problem. They also are used in the community of D -spline approximation (see [2]), where exact interpolants cannot be computed and are approximated with finite elements of a smooth class.

Unlike the other popular \mathcal{C}^1 element, the Bogner–Fox–Schmidt rectangle (that only works on grids of well orientated rectangles), the Argyris element can be used in any triangulation. Its twenty–one degrees of freedom may seem disadvantageous at first, but the fact the space involved is the full space of bivariate polynomials of degree up to five provide a very high order of convergence as the size of the triangulation decreases. However, the main drawback is the difficulty of implementation, a complication that is shared with most finite elements of Hermite type. The difficulties come from two different sources. The first one is the fact that the use of normal vectors to define degrees of freedom is not respected by affine transformations, and mapping to a reference triangle is the most widely employed way of working with triangular finite elements of any type. Therefore, Lagrange bases (in the sense of bases of the space that cancel all degrees of freedom but one) are not naturally mapped to each other. In already classical work in plate and shell approximation by Bernadou (see [4]), an exact Lagrange basis is written in terms of barycentric coordinates for an arbitrary triangle. However, this kind of treatment gives rise to the second inherent problem: to make the basis relatively simple to use, Hermitian degrees of freedom on the vertices are written in terms of directional derivatives with respect to the sides of the triangle. Hence, the local degrees of freedom of a vertex of the triangulation are not shared by all the triangles that have the vertex in common and, therefore, the global Lagrange basis of the finite element space is not the local Lagrange basis when restricted to a triangle.

The Bell element is an alternative to be considered if one wants a \mathcal{C}^1 piecewise polynomial space on a triangular grid. Constructed from the Argyris element by removing the restriction on the normal derivatives on the midpoint of each sides, it has eighteen degrees of freedom, three less than the Argyris element. However, the price paid for this simplification is that the discrete space, a particular subspace of the polynomials of degree five, is no so simple. In particular, the difficulty of manipulation of the finite element space is increased when compared with the Argyris element. Moreover, the order of approximation of the space is reduced in one.

In this work we develop a simple implementation, especially adapted to Matlab (but easily modifiable to Fortran 90), to construct and evaluate the local basis functions of the Argyris element on any triangle. The degrees of freedom on vertices will not take into account the directions of the sides and, therefore, the global basis functions of the corresponding finite element space are just constructed by joining basis functions on adjacent triangles. The main novelty that permits such an easy implementation is noticing that the matrix that transforms the basis functions from the reference triangle to a general triangle can be factored as the product of two simple rectangular matrices. The basis functions of the reference triangle can be computed with a symbolic algebra package: here we show how to do it with the symbolic package of Matlab.

We finally validate the code with two examples. First we interpolate a polynomial of degree five to see the exact matching and to validate the evaluation of the functions and their derivatives (up to order two). Then we give an example of a clamped Kirchhoff plate with exact polynomial solution.

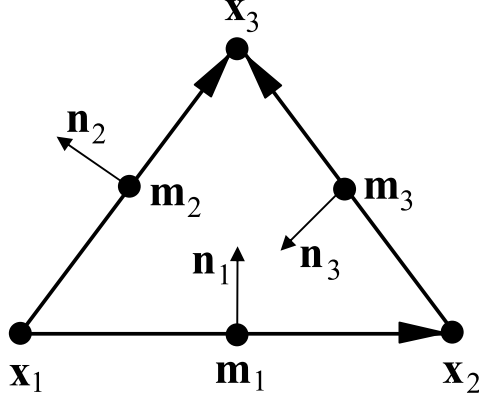


Figure 1: Geometrical elements of an arbitrary triangle

2 The Argyris element

Consider the reference triangle \widehat{K} with vertices on $\widehat{\mathbf{x}}_1 = (0, 0)$, $\widehat{\mathbf{x}}_2 = (1, 0)$ and $\widehat{\mathbf{x}}_3 = (0, 1)$. Given three non-aligned points of the plane $\mathbf{x}_\alpha = (x_\alpha, y_\alpha)$ ($\alpha = 1, 2, 3$), we consider the affine transformation that maps \widehat{K} bijectively onto K , the triangle formed by these points, $F : \widehat{K} \rightarrow K$

$$F(\mathbf{x}) = B\mathbf{x} + \mathbf{b} := \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}. \quad (1)$$

The vectors joining the vertices will be numbered in the following way (see Figure 1):

$$\mathbf{v}_1 = \mathbf{x}_2 - \mathbf{x}_1, \quad \mathbf{v}_2 = \mathbf{x}_3 - \mathbf{x}_1, \quad \mathbf{v}_3 = \mathbf{x}_3 - \mathbf{x}_2. \quad (2)$$

Also, \mathbf{n}_α denotes the unit normal vector to the corresponding side, obtained (after normalization) by rotating the vector \mathbf{v}_α an angle of $\pi/2$ in the positive direction. Finally \mathbf{m}_α will be the midpoint of the side, respecting the same numbering as above. We note that any other numbering of sides (some are more common in part of the finite element literature) amounts to a simple reordering of what follows. Also, taking the normals pointing outwards amounts only to changing some signs. We will stick to these notations, since some of the forthcoming computations become simpler.

We consider the functionals associated to the vertices of K (see Figure 2)

$$\begin{aligned} \mathcal{L}_\alpha^0(\phi) &:= \phi(\mathbf{x}_\alpha), \\ \mathcal{L}_\alpha^\circ(\phi) &:= \partial_\circ \phi(\mathbf{x}_\alpha), \quad \circ \in \{x, y\}, \\ \mathcal{L}_\alpha^\circ(\phi) &:= \partial_\circ^2 \phi(\mathbf{x}_\alpha), \quad \circ \in \{xx, xy, yy\}, \end{aligned}$$

for $\alpha \in \{1, 2, 3\}$, and the functionals associated to the sides

$$\mathcal{L}_\alpha^n(\phi) := \nabla \phi(\mathbf{m}_\alpha) \cdot \mathbf{n}_\alpha, \quad \alpha \in \{1, 2, 3\}.$$

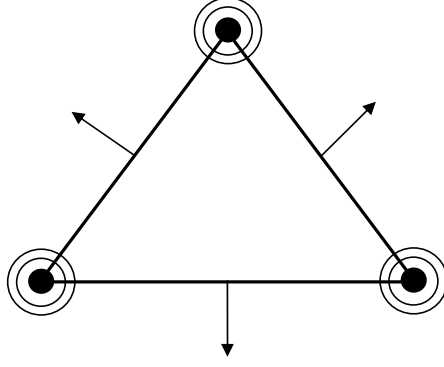


Figure 2: The usual graphical representation of the degrees of freedom of the Argyris triangle

We list these functionals as \mathcal{L}_j for $j = 1, \dots, 21$ in the following order:

$$\begin{aligned} &\mathcal{L}_1^0, \mathcal{L}_2^0, \mathcal{L}_3^0 \\ &\mathcal{L}_1^x, \mathcal{L}_1^y, \mathcal{L}_2^x, \mathcal{L}_2^y, \mathcal{L}_3^x, \mathcal{L}_3^y \\ &\mathcal{L}_1^{xx}, \mathcal{L}_1^{xy}, \mathcal{L}_1^{yy}, \mathcal{L}_2^{xx}, \mathcal{L}_2^{xy}, \mathcal{L}_2^{yy}, \mathcal{L}_3^{xx}, \mathcal{L}_3^{xy}, \mathcal{L}_3^{yy}, \\ &\mathcal{L}_1^n, \mathcal{L}_2^n, \mathcal{L}_3^n \end{aligned}$$

The functionals $\widehat{\mathcal{L}}_\alpha^\circ$ and $\widehat{\mathcal{L}}_\alpha^n$ are the corresponding on the reference triangle.

The basis functions for the Argyris element are the unique elements of \mathbb{P}_5 , the space of bivariate polynomials of degree up to five, such that

$$\mathcal{L}_i(N_j) = \delta_{ij}, \quad i, j \in \{1, \dots, 21\}.$$

Likewise, we consider the unique $\widehat{N}_j \in \mathbb{P}_5$ such that

$$\widehat{\mathcal{L}}_i(\widehat{N}_j) = \delta_{ij}, \quad i, j \in \{1, \dots, 21\}.$$

We define

$$\widetilde{\mathcal{L}}_i(\phi) := \widehat{\mathcal{L}}_i(\phi \circ F)$$

i.e. $\widetilde{\mathcal{L}}_i(\phi \circ F^{-1}) := \widehat{\mathcal{L}}_i(\phi)$. Since both sets $\{\mathcal{L}_i\}$ and $\{\widetilde{\mathcal{L}}_i\}$ are bases of \mathbb{P}_5^* , the dual space to \mathbb{P}_5 , there exists a matrix $C = (c_{ij})$ such that

$$\widetilde{\mathcal{L}}_i = \sum_{j=1}^{21} c_{ij} \mathcal{L}_j, \quad \text{in } \mathbb{P}_5^*, \quad i = 1, \dots, 21. \quad (3)$$

By an elementary transposition argument, it follows that

$$N_i \circ F = \sum_{k=1}^{21} c_{ki} \widehat{N}_k, \quad i = 1, \dots, 21.$$

The aim of the following section is giving a simple expression for the matrix C .

3 Computation of the change of bases

We consider the matrix

$$R = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

which rotates vectors an angle $\pi/2$ in the positive sense. Next, we introduce a new set of functionals

$$\mathcal{L}_i^*, \quad i = 1, \dots, 24$$

doing as follows. We keep the vertex-associated functionals $\mathcal{L}_i^* := \mathcal{L}_i$, $i = 1, \dots, 18$ in their original order. We then introduce the functionals

$$\mathcal{L}_\alpha^\circ, \quad \alpha \in \{1, 2, 3\}, \quad \circ \in \{\perp, \parallel\}$$

by the expressions

$$\mathcal{L}_\alpha^\perp(\phi) := \nabla\phi(\mathbf{m}_\alpha) \cdot R\mathbf{v}_\alpha, \quad \mathcal{L}_\alpha^\parallel(\phi) := \nabla\phi(\mathbf{m}_\alpha) \cdot \mathbf{v}_\alpha$$

ordered in the following way: $\mathcal{L}_1^\perp, \mathcal{L}_2^\perp, \mathcal{L}_3^\perp, \mathcal{L}_1^\parallel, \mathcal{L}_2^\parallel, \mathcal{L}_3^\parallel$.

If we write gradients columnwise $\nabla\phi := (\partial_x\phi, \partial_y\phi)^\top$, then

$$\nabla(\phi \circ F) = B^\top \nabla\phi \circ F.$$

Also, by using the following column form of the hessian matrix $\mathcal{H}(\phi) := (\partial_{xx}\phi, \partial_{xy}\phi, \partial_{yy}\phi)^\top$, it follows that

$$\mathcal{H}(\phi \circ F) = \Theta \mathcal{H}(\phi) \circ F$$

where

$$\Theta := \begin{bmatrix} b_{11}^2 & 2b_{11}b_{21} & b_{21}^2 \\ b_{12}b_{11} & b_{12}b_{21} + b_{11}b_{22} & b_{21}b_{22} \\ b_{12}^2 & 2b_{22}b_{12} & b_{22}^2 \end{bmatrix}.$$

We employ these matrices to construct a 21×24 matrix D in block diagonal form:

$$D := \text{diag}(I_3, B^\top, B^\top, B^\top, \Theta, \Theta, \Theta, Q)$$

where I_3 is a 3×3 identity matrix and the block Q is a 3×6 matrix

$$Q := \left[\begin{array}{ccc|ccc} f_1 & & & g_1 & & \\ & f_2 & & & g_2 & \\ & & f_3 & & & g_3 \end{array} \right],$$

constructed as follows. If $\ell_\alpha := |\mathbf{v}_\alpha|$ is the length of the α th side and

$$\begin{bmatrix} \mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \mathbf{a}_3^\top \end{bmatrix} := \begin{bmatrix} \ell_1^{-2} & & \\ & \ell_2^{-2} & \\ & & \ell_3^{-2} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \\ -1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} B^\top, \quad (4)$$

then

$$f_\alpha := \mathbf{a}_\alpha \cdot R\mathbf{v}_\alpha, \quad g_\alpha := \mathbf{a}_\alpha \cdot \mathbf{v}_\alpha, \quad \alpha \in \{1, 2, 3\}. \quad (5)$$

The diag instruction corresponds to the Matlab `blkdiag` command and creates a matrix by adding each block to the right-lower-most corner of the matrix created by the preceding blocks.

Proposition 1 *If \mathbb{P} is the space of bivariate polynomials of arbitrary order and \mathbb{P}^* is its dual space, then*

$$\tilde{\mathcal{L}}_i = \sum_{j=1}^{24} d_{ij} \mathcal{L}_j^*, \quad \text{in } \mathbb{P}^*, \quad i = 1, \dots, 21. \quad (6)$$

Proof. We first remark that

$$\tilde{\mathcal{L}}_\alpha^0 = \mathcal{L}_\alpha^0, \quad \begin{bmatrix} \tilde{\mathcal{L}}_\alpha^x \\ \tilde{\mathcal{L}}_\alpha^y \end{bmatrix} = B^\top \begin{bmatrix} \mathcal{L}_\alpha^x \\ \mathcal{L}_\alpha^y \end{bmatrix}, \quad \begin{bmatrix} \tilde{\mathcal{L}}_\alpha^{xx} \\ \tilde{\mathcal{L}}_\alpha^{xy} \\ \tilde{\mathcal{L}}_\alpha^{yy} \end{bmatrix} = \Theta \begin{bmatrix} \mathcal{L}_\alpha^{xx} \\ \mathcal{L}_\alpha^{xy} \\ \mathcal{L}_\alpha^{yy} \end{bmatrix}.$$

Notice that $\mathbf{v}_\alpha = B \hat{\mathbf{v}}_\alpha$ for $\alpha \in \{1, 2, 3\}$, where $\hat{\mathbf{v}}_\alpha$ are defined in the reference triangle as in (2). Then

$$\tilde{\mathcal{L}}_\alpha^n = \frac{1}{|\hat{\mathbf{v}}_\alpha|} R \hat{\mathbf{v}}_\alpha \cdot \ell_\alpha^{-2} B^\top \begin{bmatrix} -v_\alpha^y & v_\alpha^x \\ v_\alpha^x & v_\alpha^y \end{bmatrix} \begin{bmatrix} \mathcal{L}_{\alpha\beta}^\perp \\ \mathcal{L}_{\alpha\beta}'' \end{bmatrix} \quad (7)$$

To prove (7) we just have to notice that

$$\tilde{\mathcal{L}}_\alpha^n(\phi) = \hat{\mathcal{L}}_\alpha^n(\phi \circ F) = \frac{1}{|\hat{\mathbf{v}}_\alpha|} R \hat{\mathbf{v}}_\alpha \cdot \nabla(\phi \circ F)(\hat{\mathbf{m}}_\alpha) = \frac{1}{|\hat{\mathbf{v}}_\alpha|} R \hat{\mathbf{v}}_\alpha \cdot B^\top \nabla \phi(\mathbf{m}_\alpha)$$

and that, since

$$\begin{bmatrix} \mathcal{L}_\alpha^\perp \\ \mathcal{L}_\alpha'' \end{bmatrix} = \begin{bmatrix} -v_\alpha^y & v_\alpha^x \\ v_\alpha^x & v_\alpha^y \end{bmatrix} \begin{bmatrix} \mathcal{L}_\alpha^x \\ \mathcal{L}_\alpha^y \end{bmatrix}$$

then

$$\begin{bmatrix} \mathcal{L}_\alpha^x \\ \mathcal{L}_\alpha^y \end{bmatrix} = \frac{1}{|\mathbf{v}_\alpha|^2} \begin{bmatrix} -v_\alpha^y & v_\alpha^x \\ v_\alpha^x & v_\alpha^y \end{bmatrix} \begin{bmatrix} \mathcal{L}_\alpha^\perp \\ \mathcal{L}_\alpha'' \end{bmatrix}.$$

Finally, it follows readily that (7) can be also read as

$$\tilde{\mathcal{L}}_\alpha^n = f_\alpha \mathcal{L}_\alpha^\perp + g_\alpha \mathcal{L}_\alpha'',$$

with

$$f_\alpha = \frac{1}{\ell_\alpha^2 |\hat{\mathbf{v}}_\alpha|} R \hat{\mathbf{v}}_\alpha \cdot B^\top R \mathbf{v}_\alpha, \quad g_\alpha = \frac{1}{\ell_\alpha^2 |\hat{\mathbf{v}}_\alpha|} R \hat{\mathbf{v}}_\alpha \cdot B^\top \mathbf{v}_\alpha$$

and that these expressions of f_α and g_α correspond to the ones given in the construction of D . \square

Consider now a new matrix E , 24×21 , with the following structure

$$E = \begin{bmatrix} I_{18} & 0 \\ 0 & L \\ T & 0 \end{bmatrix}, \quad L = \text{diag}(\ell_1, \ell_2, \ell_3),$$

where the last block (3×18) is composed by concatenation of three subblocks (3×3 , 3×6 and 3×9 respectively)

$$\frac{15}{18} \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix}, \quad -\frac{7}{16} \begin{bmatrix} \mathbf{v}_1^\top & \mathbf{v}_1^\top & \mathbf{0} \\ \mathbf{v}_2^\top & \mathbf{0} & \mathbf{v}_2^\top \\ \mathbf{0} & \mathbf{v}_3^\top & \mathbf{v}_3^\top \end{bmatrix}, \quad \frac{1}{32} \begin{bmatrix} -\mathbf{w}_1^\top & \mathbf{w}_1^\top & \mathbf{0} \\ -\mathbf{w}_2^\top & \mathbf{0} & \mathbf{w}_2^\top \\ \mathbf{0} & -\mathbf{w}_3^\top & \mathbf{w}_3^\top \end{bmatrix},$$

with $\mathbf{w}_\alpha^\top = (v_\alpha^x v_\alpha^x, 2v_\alpha^x v_\alpha^y, v_\alpha^y v_\alpha^y)$.

Proposition 2 *It holds*

$$\mathcal{L}_i^* = \sum_{j=1}^{21} e_{ij} \mathcal{L}_j, \quad \text{in } \mathbb{P}_5^*, \quad i = 1, \dots, 24. \quad (8)$$

Proof. It is clear that $\mathcal{L}_j^* = \mathcal{L}_j$ for $j = 1, \dots, 18$ and that since $\ell_\alpha \mathbf{n}_\alpha = R\mathbf{v}_\alpha$, then $\mathcal{L}_\alpha^\perp = \ell_\alpha \mathcal{L}_\alpha^n$. Take now an arbitrary $\phi \in \mathbb{P}_5$ and define

$$\psi(t) := \phi(t\mathbf{x}_\beta + (1-t)\mathbf{x}_\alpha) \in \mathbb{P}_5(t), \quad \alpha < \beta.$$

Notice that for all $\psi \in \mathbb{P}_5(t)$,

$$\psi'(1/2) = \frac{15}{8}(\psi(1) - \psi(0)) - \frac{7}{16}(\psi'(1) + \psi'(0)) + \frac{1}{32}(\psi''(1) - \psi''(0)). \quad (9)$$

Take now the index γ so that $\mathbf{v}_\gamma = \mathbf{x}_\beta - \mathbf{x}_\alpha$ (see (2) for the correspondence). Hence

$$\begin{aligned} \psi'(1/2) &= \mathcal{L}_\gamma^{\text{I}}(\phi) \\ \psi(0) &= \mathcal{L}_\alpha^0(\phi) \\ \psi(1) &= \mathcal{L}_\beta^0(\phi) \\ \psi'(0) &= v_\gamma^x \mathcal{L}_\alpha^x(\phi) + v_\gamma^y \mathcal{L}_\alpha^y(\phi) \\ \psi'(1) &= v_\gamma^x \mathcal{L}_\beta^x(\phi) + v_\gamma^y \mathcal{L}_\beta^y(\phi) \\ \psi''(0) &= (v_\gamma^x)^2 \mathcal{L}_\alpha^{xx}(\phi) + 2v_\gamma^x v_\gamma^y \mathcal{L}_\alpha^{xy}(\phi) + (v_\gamma^y)^2 \mathcal{L}_\alpha^{yy}(\phi) \\ \psi''(1) &= (v_\gamma^x)^2 \mathcal{L}_\beta^{xx}(\phi) + 2v_\gamma^x v_\gamma^y \mathcal{L}_\beta^{xy}(\phi) + (v_\gamma^y)^2 \mathcal{L}_\beta^{yy}(\phi). \end{aligned}$$

Applying (9) we obtain the following expression

$$\begin{aligned} \mathcal{L}_\gamma^{\text{I}} &= \frac{15}{8}(-\mathcal{L}_\alpha^0 + \mathcal{L}_\beta^0) - \frac{7}{16} \left(v_\gamma^x \mathcal{L}_\alpha^x + v_\gamma^y \mathcal{L}_\alpha^y + v_\gamma^x \mathcal{L}_\beta^x + v_\gamma^y \mathcal{L}_\beta^y \right) \\ &\quad + \frac{1}{32} \left(-(v_\gamma^x)^2 \mathcal{L}_\alpha^{xx} - 2v_\gamma^x v_\gamma^y \mathcal{L}_\alpha^{xy} - (v_\gamma^y)^2 \mathcal{L}_\alpha^{yy} + (v_\gamma^x)^2 \mathcal{L}_\beta^{xx} + 2v_\gamma^x v_\gamma^y \mathcal{L}_\beta^{xy} + (v_\gamma^y)^2 \mathcal{L}_\beta^{yy} \right). \end{aligned}$$

This identity finishes the proof. \square

Notice that, combining (6) and (8) we obtain that the matrix in (3) is simply

$$C = D E. \quad (10)$$

This gives an expression of the change of bases from the reference element to any triangle factored as the product of two rectangular very sparse matrices constructed with simple geometric elements of the triangle. In the next section we give Matlab code for the construction of C as well as its use to evaluate the basis functions N_i as well as their derivatives of first and second order.

4 Documentation and implementation

The package consist of the following routines. First we have an initialisation subroutine

- **reference**: computes the basis of the Argyris element in the reference triangle. The result is stored in a global variable **M**. Variable **M** is a 21×21 matrix, $M(i, :)$ being the coefficients of \widehat{N}_i in the monomial basis.

We emphasise that this process must be done once. The function makes use of the symbolic toolbox of Matlab. Since the only purpose of this function is to calculate **M** and save it as a global variable, it is possible to compute the coefficients in a different manner, for instance by means of a symbolic processor, and store the result in a data file.

The main files of the package are listed below

- **z=evalArgy(x,y,p)**: **x**, **y** are scalars (or row vectors of the same length) with the coordinates of a point (or a set of points) where we will evaluate the basis functions; **p** is a 2×3 matrix that determines the triangle. Its j th column are the coordinates of the j th vertex. The only output argument (**z**) is a $21 \times k$ matrix k being the length of **x** or **y**, where the i th row is the values of the i th element of the basis at the points given by **x**, **y**.
- **[dx,dy]=evalGradArgy(x,y,p)** returns in **dx,dy** the values of the x - and y - derivatives of the elements of the local basis of the Argyris element at (x, y) . All the input and output arguments follow the convention above.
- **[dxx,dxy,dyy]=evalHessArgy(x,y,p)** evaluates the second derivatives of the elements of the Argyris basis at (x, y) .

Finally we have some auxiliary routines used in the computations:

- **[C,B,b,Th]=changeOfBasis(p)** computes C , B , **b** and Θ , the matrices (and vector) involved in the changes of coordinates between the reference triangle and the triangle given by **p**.
- **khat2k, k2khat** computes the image and the inverse image of the affine mapping (1).

In the remainder of the section we give some details about the implementation of this package, starting with the function **changeOfBasis**. The computation of B , **b** and Θ is straightforward and is made in an internal function called **afftrans**. The bulk of the code is devoted to the computation of C . We distinguish three parts. The first part defines some geometric quantities

```
v=[p(:,2)- p(:,1), p(:,3)-p(:,1), p(:,3)-p(:,2)];
[B,b,Th]=afftrans(p);
sides=diag([norm(v(:,1)),norm(v(:,2)),norm(v(:,3))]);
aux=sides^(-2)*[0 1; -1 0; -1/sqrt(2) -1/sqrt(2)]*B'; % see (3)
R=[0 -1; 1 0];
```

Next we construct the matrix D :

```
f=dot(aux',v);
g=dot(aux',R*v);
D=blkdiag(eye(3),B',B',B',Th,Th,Th,[diag(g) diag(f)]);
```

Note the use of the Matlab command `dot` to compute f_α and g_α by doing the dot product between the columns of appropriate matrices (see (5)). The command `blkdiag` is finally employed to assembly the matrix D .

The construction of the matrix E is done similarly

```
E=zeros(24,21);
E(1:21,:)=blkdiag(eye(18),sides);
E(22:24,1:3)= 15/8*[-1  1 0;  -1  0 1;  0 -1 1];
E(22:24,4:9)=-7/16*[v(:,1) v(:,1)  0  0;
                    v(:,2)  0  0  v(:,2)'];...
                    0  0  v(:,3) v(:,3)'];
w=[v(1,:).^2; 2*v(1,:).*v(2,:); v(2,:).^2]';
E(22:24,10:18)=1/32*[-w(1,:)  w(1,:)  0  0  0;.
                    -w(2,:)  0  0  0  w(2,:);...
                    0  0  0  -w(3,:)  w(3,:)];
```

The program finishes by computing C

```
C=D*E;
```

The evaluation of the Argyris basis in the triangle specified by \mathbf{p} is done in the following lines

```
[C,B,b]=changeOfBasis(p);
[x,y]=k2khat(x,y,B,b);
z=monomials(x,y);
z=C'*M*z;
```

The point (x, y) is mapped first into the reference triangle and next the elements of the monomial basis are evaluated at this point (note that the i th row of $\mathbf{M} \cdot \mathbf{z}$ corresponds to $\widehat{N}_i(x, y)$). The change of basis, and therefore the evaluation of the local basis $N_i(x, y)$ in the user-specified triangle, is carried out by the left multiplication by \mathbf{C}' .

This code can be vectorized just by allowing both \mathbf{x}, \mathbf{y} to be row vectors of the same length. If k is the length of $\mathbf{x}, \mathbf{y}, \mathbf{z}$, becomes a $21 \times k$ matrix in all the occurrences.

To compute the first derivatives we use the chain rule (recall that the gradient is seen columnwise)

$$(\nabla N_j(\mathbf{x}))^\top = \sum_{i=1}^n c_{ij} (\nabla \widehat{N}_i)^\top \circ F^{-1}(\mathbf{x}) B^{-1}$$

This is done in the following lines

```
[C,B,b]=changeOfBasis(p);
[x,y]=k2khat(x,y,B,b);
k=length(x);
mx=derx(x,y);
my=dery(x,y);
grads=zeros(21,2*k);
grads(:)=[mx(:) my(:)]*inv(B);
```

```

grads=C'*M*grads;
dx=grads(:,1:k);
dy=grads(:,k+1:2*k);

```

Functions `derx`, `dery` return a column vector with the derivatives of the monomial basis evaluated at (x, y) . The columnwise access to the elements of a matrix in Matlab is used here to set `grads` in such a way that after running the six first lines, `grads` has in the first k columns $\partial_x(m_i \circ F^{-1})(x_j, y_j)$ (here m_i denotes the i th element of the basis of monomials and F the affine mapping from \widehat{K} onto K) whereas $\partial_y(m_i \circ F^{-1})(x_j, y_j)$ are stored in the last k columns. Finally, left multiplication by `M` and `C'` makes the change of basis.

The evaluation of the second derivatives is implemented in the same manner:

```

[C,B,b,Th]=changeOfBasis(p);
[x,y]=k2khat(x,y,B,b);
k=length(x);
mxx=derxx(x,y);
mxy=derxy(x,y);
myy=deryy(x,y);
hess=zeros(21,3*k);
hess(:)=[mxx(:) mxy(:) myy(:)]*inv(Th');
hess=C'*M*hess;
dxx=hess(:,1:k);
dxy=hess(:,k+1:2*k);
dyy=hess(:,2*k+1:3*k);

```

5 Numerical tests

The aim of this section is to validate the code by developing two applications that use it.

The first experiment concerns the interpolation of a function on an arbitrary triangle. That is, given a triangle K we define for any function f smooth enough the interpolation operator

$$\mathbb{P}_5 \ni p \quad \text{s.t.} \quad \mathcal{L}_j(p - f) = 0, \quad j = 1, \dots, 21.$$

In our test we have taken $f(x, y) = x^5 - x^4y + 2xy - 3x^2y^2$ on the triangle K with vertices $\{(1, 0), (-1/2, \sqrt{2}/2), (-1/2, -\sqrt{2}/2)\}$. Since f is a polynomial of degree 5, the error is zero in exact arithmetic.

We construct the grid resulting of six consecutive uniform refinements of the original triangle. As a L^∞ error estimate we introduce

$$\begin{aligned}
E &:= \max_{(x,y) \in \mathcal{E}} |f(x, y) - p(x, y)|, \\
E_\circ &:= \max_{(x,y) \in \mathcal{E}} |\partial_\circ f(x, y) - \partial_\circ p(x, y)|, \quad \circ \in \{x, y, xx, xy, yy\},
\end{aligned}$$

\mathcal{E} being the set of nodes of the finest grid. The errors are given in Table 1

In the next experiment we have implemented the finite element solution of the Dirichlet problem for the biharmonic equation. Let Ω be the regular hexagon inscribed in the unit

E	E_x	E_y	E_{xx}	E_{xy}	E_{yy}
1.93E - 14	4.66E - 14	5.51E - 14	7.73E - 14	6.36E - 14	1.29E - 13

Table 1: L^∞ error estimate of the evaluation of the Argyris basis

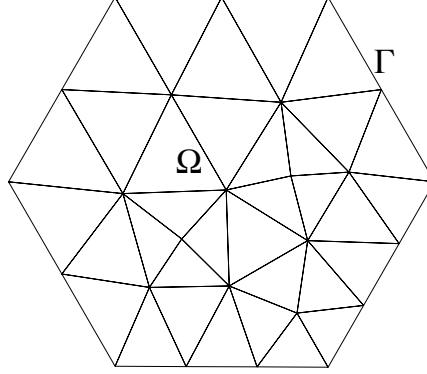


Figure 3: Domain of the differential problem and initial grid

circle and \mathcal{T}_h , the set of triangles of the grid depicted in Figure 3. This initial grid consists of 34 triangles. On \mathcal{T}_h and on successively uniform refinements of it, we construct the \mathcal{C}^1 finite element spaces

$$V_h := \left\{ u_h \in C^1(\Omega) \mid u_h|_T \in \mathbb{P}_5, \forall T \in \mathcal{T}_h \right\}, \quad V_h^0 := \left\{ u_h \in V_h \mid \partial_\nu u_h = \gamma_\Gamma u_h = 0 \right\}.$$

Here γ_Γ and ∂_ν denote the trace and the outward normal derivative on Γ , the boundary of Ω . We consider the variational problem

$$u_h \in V_h^0, \quad \text{s.t.} \quad a(u_h, v_h) = \int_\Omega f v_h, \quad \forall v_h \in V_h^0, \quad (11)$$

with

$$a(u, v) := \int_\Omega (\partial_{xx}^2 u \partial_{xx}^2 v + 2 \partial_{xy}^2 u \partial_{xy}^2 v + \partial_{yy}^2 u \partial_{yy}^2 v).$$

The solution u_h of the numerical scheme is the finite element solution of the boundary value problem

$$\Delta^2 u = f, \quad \text{in } \Omega, \quad \gamma_\Gamma u = 0, \quad \partial_\nu u = 0. \quad (12)$$

In our test, the right-hand-side f is taken so that

$$u(x, y) = \prod_{i=1}^6 (\alpha_i + \beta_i x - y)^2$$

\mathcal{T}_h^0	\mathcal{T}_h^1	\mathcal{T}_h^2	\mathcal{T}_h^3
4.49E - 02	3.94E - 04	1.01E - 05	2.47E - 07

Table 2:

is the exact solution of (12). In this expression, $y = \alpha_i + \beta_i x$ is the equation of the line containing the i th side of the hexagon.

The assembly of the matrix is done element by element in the usual finite element way. This is possible, as we remark in the introduction, because for any triangle the elements of the global basis of V_h whose support has non trivial intersection with K give the local Argyris basis of this triangle.

The assembly of the matrix and of the right-hand-side requires the computations of some integrals involving the elements of the local basis of the Argyris element on each triangle and their second derivatives. These integrals are computed by using a Gaussian quadrature formula of degree 6 (which is exact for the entries of the matrix of (11)). This quadrature formula uses a very high number of evaluations of the integrand per triangle (25 points in our implementation). Hence, we have a practical example where we can test the performance of our algorithms when they are put in the context of a practical problem. In our experiments the profile tool of Matlab has been used to track the performance of the different functions involved. We have observed that the evaluation of the Argyris functions consumes a minor fraction of the CPU time spent in the assembly of the matrix. Moreover, the finest the grid is, the less the percentage of the CPU time used in the evaluation becomes. In any case the bulk of the computational time is spent when the local contributions of each triangle are transferred to the corresponding entries of the matrix.

Table 2 shows an estimate of the L^∞ error for different uniform refinements of \mathcal{T}_h (\mathcal{T}_h^j is the grid obtained after j steps of uniform refinement).

Acknowledgements

The authors are partially supported by FEDER/MCYT Projects MTM2004-01905, DGA (Grupo consolidado PDIE) and Gobierno de Navarra, resolución 18/2005

References

- [1] J.H. Argyris, I. Fried, and D.W. Scharpf, *The TUBA family of plate elements for the matrix displacement method*, Aero. J. Roy. Aero. Soc **72** (1968), 701–709.
- [2] R. Arcangéli, M. Cruz López de Silanes and J.J. Torrens, *Multidimensional minimizing splines*, Grenoble Sciences, Kluwer, 2004.
- [3] K. Bell, *A refined triangular plate bending finite element*, Internat. J. Numer. Methods Engrg. **1** (1969), 101–122.

- [4] M. Bernadou, *Méthodes d'éléments finis pour les problèmes de coques minces*, Dunod, 1997.
- [5] D. Braess, *Finite elements*, second ed., Cambridge University Press, Cambridge, 2001.
- [6] S.C. Brenner and L.R. Scott, *The mathematical theory of finite element methods*, Springer-Verlag, 2002.
- [7] P.G. Ciarlet, *The finite element method for elliptic problems*, North Holland, 1975.