Analysis of Thin Shells by the Element-Free Galerkin Method

Petr Krysl and Ted Belytschko*

1996

Abstract

A meshless approach to the analysis of arbitrary Kirchhoff shells by the Element-Free Galerkin (EFG) method is presented. The shell theory used is geometrically exact and can be applied to deep shells. The method is based on moving least squares approximant. The method is meshless, which means that the discretization is independent of the geometric subdivision into "finite elements". The satisfaction of the C^1 continuity requirements are easily met by EFG since it requires only C^1 weights; therefore, it is not necessary to resort to Mindlin-Reissner theory or to devices such as discrete Kirchhoff theory. The requirements of consistency are met by the use of a polynomial basis of quadratic or higher order. A subdivision similar to finite elements is used to provide a background mesh for numerical integration. The essential boundary conditions are enforced by Lagrange multipliers. Membrane locking, which is due to different approximation order for transverse and membrane displacements, is removed by using larger domains of influence with the quadratic basis, and by using quartic polynomial basis, which can prevent membrane locking completely. It is shown on the obstacle course for shells that the present technique performs well.

1 Introduction

There is a growing interest in the so-called "meshless" methods, particularly for problems involving continuous changes in geometry such as dynamic fracture. It might be partly traced to high costs involved in meshing procedures. These problems involve considerable remeshing efforts, which can easily constitute the largest portion of analysis costs. Meshless methods do not require a finite element mesh for the definition of the approximation. The discretization is based on a set of nodes (ordered or scattered), although a background mesh may be used for quadrature. The connectivity

*Department of Civil Engineering, Robert R. McCormick School of Engineering and Applied Science, The Technological Institute, Northwestern University, Evanston, IL 60208-3109, U.S.A.

in terms of node interactions may be changing constantly, and modelling of fracture, free surfaces, large deformations, etc. is considerably simplified – *cf.* Belytschko *et al.* (1994) [1].

Meshless methods have been proposed in several varieties as Generalized Finite Difference Method (Liszka (1984) [14]), Smoothed Particle Hydrodynamics (Monaghan (1982) [17]), Diffuse Element Method (Nayroles (1992) [18]), Wavelet Galerkin Method (e.g. Qian and Weiss (1993) [19]), Multiquadrics (Kansa (1990) [9, 10]), Reproducing Kernel Particle Methods (Liu *et al.* (1995) [15]) and the Element-Free Galerkin Method (Belytschko *et al.* (1994) [2]).

The Element-Free Galerkin Method (EFGM) is based on a moving least squares approximation. These approximations originated in scattered data fitting, where it has been studied under different names (local regression, "loess", and moving least squares) since the 1920's – *cf.* Cleveland (1993) [7] and Lancaster and Šalkauskas (1986) [13].

The enforcement of essential boundary conditions in the EFGM requires special treatment, therefore a number of techniques have been proposed such as point collocation, Lagrange multipliers, and coupling with finite elements. The coupling with finite element methods seems especially desirable as the computational costs are relatively high for the EFG method due to its dynamic connectivity character (the connectivity, i.e. the interaction of nodes, is not fixed by input data, it needs to be computed), and it is anticipated that EFG would be used only where fracture is expected; alternatively a transition from finite elements to EFG in areas of fracture could be used.

The goal of the present paper is to develop and study the EFG method for problems of thin shells – usually denoted as Kirchhoff shells. The problem of constructing C^1 finite elements for shells of general shape has been addressed by many researchers. Although C^1 elements have been developed, alternative methodologies which circumvent the continuity requirement seem to have become predominant in recent years. The most popular C^0-type methods are those based on Mindlin-Reissner shell theory, and the hybrid and mixed models.

The EFG method offers considerable potential with respect to numerical solutions of boundary-value problems that require high continuity in the trial functions – Kirchhoff shell theory being one of them. The continuity of the shape functions is primarily governed by the continuity of the weight function. Therefore, as it is possible to construct sufficiently smooth weight functions, the numerical approach is greatly simplified.

An earlier paper by the authors dealt with the EFG method for thin plates. High performance and insensitivity to grid irregularity have been demonstrated. Numerical studies to assess the influence of the support size of the weight functions on the accuracy, and the required quadrature order were presented. These results have been applied in this study in shell problems.

The outline of the paper is as follows: First, a very short account of the numerical formulation of the Kirchhoff shell theory is given. The EFG method approximation

is then reviewed: the moving least squares technique, the properties of the EFG approximation, and the construction of the shape functions. The discretization issues are then discussed: The surface approximation techniques , surface approximation quality issues, algorithm for automatic parameterization of the surface, choice of displacement parameters, computation of the stiffness matrix, and the enforcement of the essential boundary conditions. A discussion of the choice of the weight function then follows, with some comments on the way in which the choice of the weight function support affects the solution. Next section discusses the phenomenon of membrane locking, and devices to alleviate it are proposed.

The paper is concluded by a section on numerical experiments. The well-known shell benchmarks from the shell obstacle course of Belytschko *et al.* [4] are applied.

2 Governing equations

The shells, considered in the present work, are assumed to be thin so that the Kirchhoff-Love theory can be considered appropriate, and arbitrarily deep with any Gaussian curvature. The formulation of the governing equations used in this report is based on a series of papers on geometrically exact theory of shear flexible shells by Simo et al. – *cf.* papers [20, 21], and appropriate adjustements were made to account for the Kirchhoff hypothesis. However, the hypothesis is invoked at the latest stages to avoid cluttering up the equations.

2.1 Kinematic description of shell

The Gauss intrinsic coordinates (a normal coordinate chart) are used to describe the configuration of the shell. The shell in the 3D space is described in a global cartesian coordinate system $\boldsymbol{E}_k$. The pair $(\boldsymbol{\varphi}, \boldsymbol{t})$ defines the position of an arbitrary point of the shell, $\boldsymbol{\varphi}$ gives the position of a point on the shell midsurface, and $\boldsymbol{t}$ is a director unit vector (normal to the shell surface both in the reference and deformed states – the usual Kirchhoff-Love hypothesis). The configuration $\mathcal{S}$ (generic state) can be put down as

$$\mathcal{S} = \left\{ \boldsymbol{x} \in R^3 \mid \boldsymbol{x} = \boldsymbol{\varphi}(\xi^1, \xi^2) + \xi \boldsymbol{t}(\xi^1, \xi^2) \text{ with } \xi^1, \xi^2 \in \mathcal{A} \text{ and } \xi \in \left\langle h^-, h^+ \right\rangle \right\}. \quad (2.1)$$

Here $\mathcal{A}$ denotes the parametric space, h^-, h^+ are the distances of the "lower" and "upper" surfaces of the shell from the reference surface. Superscript 0 denotes quantities in the reference configuration, for instance $\boldsymbol{\varphi}^0$ is a point on the reference surface.

We define the convective basis vectors $\boldsymbol{g}_I$ by the tangent map

$$\nabla \boldsymbol{x} = \frac{\partial \boldsymbol{x}}{\partial \xi^I} \otimes \boldsymbol{E}^I = \boldsymbol{g}_I \otimes \boldsymbol{E}^I \ . \quad (2.2)$$

A contravariant (dual) basis $\boldsymbol{g}^I$ can be obtained from the standard relation $\boldsymbol{g}_I \cdot \boldsymbol{g}^J = \delta_I^J$. The determinants of the tangent maps will be denoted subsequently as j and j^0

respectively (denoting $\bar{j}$ and $\bar{j}^0$ the Jacobians on the reference surface)

$$j = \det\left[\nabla \boldsymbol{x}\right], \quad j^0 = \det\left[\nabla \boldsymbol{x}^0\right] \ , \ \bar{j} = j\big|_{\xi=0}, \ \bar{j}^0 = \bar{j}\big|_{\xi=0} \ . \tag{2.3}$$

The surface differential (a two-form) is defined by

$$\mathrm{d}\mathcal{A} = \boldsymbol{\varphi}_{,1} \times \boldsymbol{\varphi}_{,2} \ \mathrm{d}\xi^1 \mathrm{d}\xi^2 \ . \tag{2.4}$$

2.2 Stress resultants and stress couples

To define the stress resultants we introduce a section through the shell at $\xi^\alpha = \text{const}$.

$$\mathcal{S}^\alpha = \left\{\boldsymbol{x} = \boldsymbol{x}|_{\xi^\alpha = \text{const}}\right\}, \ \alpha = 1, 2 \tag{2.5}$$

The one-form normal to the section surface for $\xi^1 = \text{const}$ is given by

$$\mathrm{d}\mathcal{S}^1 = j\left[\nabla \boldsymbol{x}\right]^{-t} \boldsymbol{E}^1 \mathrm{d}\xi^2 \mathrm{d}\xi = j\boldsymbol{g}^1 \mathrm{d}\xi^2 \mathrm{d}\xi \ , \tag{2.6}$$

The force acting on the section $\mathcal{S}^1$ per unit of coordinate length can be therefore written as ($\boldsymbol{\sigma}$ being the Cauchy stress tensor)

$$\boldsymbol{R}^1 = \int_{h^-}^{h^+} \boldsymbol{\sigma} \frac{\mathrm{d}\mathcal{S}^1}{\mathrm{d}\xi^2} = \int_{h^-}^{h^+} \boldsymbol{\sigma}\boldsymbol{g}^1 j \mathrm{d}\xi \ , \tag{2.7}$$

and similarly the couple acting on $\mathcal{S}^1$ per unit of coordinate length

$$\boldsymbol{T}^1 = \int_{h^-}^{h^+} (\boldsymbol{x} - \boldsymbol{\varphi}) \times \boldsymbol{\sigma} \frac{\mathrm{d}\mathcal{S}^1}{\mathrm{d}\xi^2} = \int_{h^-}^{h^+} (\boldsymbol{x} - \boldsymbol{\varphi}) \times \boldsymbol{\sigma}\boldsymbol{g}^1 j \mathrm{d}\xi \ . \tag{2.8}$$

The stress resultants (force $\boldsymbol{n}^\alpha$ and couple $\boldsymbol{m}^\alpha$) are normalized $\boldsymbol{R}^\alpha$ and $\boldsymbol{T}^\alpha$ with respect to the surface Jacobian $\bar{j}$

$$\boldsymbol{n}^\alpha = (\bar{j})^{-1} \boldsymbol{R}^\alpha \tag{2.9}$$

$$\boldsymbol{m}^\alpha = (\bar{j})^{-1} \boldsymbol{T}^\alpha \tag{2.10}$$

The stress resultant couple can also be expressed as

$$\boldsymbol{m}^\alpha = \boldsymbol{t} \times \tilde{\boldsymbol{m}}^\alpha, \ \text{with } \tilde{\boldsymbol{m}}^\alpha = (\bar{j})^{-1} \int_{h^-}^{h^+} \xi \boldsymbol{\sigma}\boldsymbol{g}^\alpha j \mathrm{d}\xi \tag{2.11}$$

The across-the-thickness resultant has been omitted as it does not play a role in the Kirchhoff-Love theory.

3 Principle of virtual work

Let us introduce the following kinematic variables (the fundamental forms of the shell surface):

$$\text{(a) 1st fundamental form:} \quad a_{\alpha\beta} = \boldsymbol{\varphi}_{,\alpha} \cdot \boldsymbol{\varphi}_{,\beta} \tag{3.1}$$

$$\text{(b) 2nd fundamental form:} \quad \kappa_{\alpha\beta} = \boldsymbol{\varphi}_{,\alpha} \cdot \boldsymbol{t}_{,\beta} \tag{3.2}$$

The static weak form can be put into the following component form in the effective resultants

$$W(\delta\boldsymbol{x}) = \int_{\mathcal{A}} \left[\tilde{n}^{\beta\alpha}{}_{\frac{1}{2}}\delta a_{\beta\alpha} + \tilde{m}^{\beta\alpha}\delta\kappa_{\beta\alpha}\right] \, \mathrm{d}\mathcal{A} - W_{ext}(\delta\boldsymbol{x}) \; , \tag{3.3}$$

with the virtual work of the external loading $W_{ext}(\delta\boldsymbol{x})$ can be expressed as

$$W_{ext}(\delta\boldsymbol{x}) = \int_{\mathcal{A}} \left[\bar{\boldsymbol{n}} \cdot \delta\boldsymbol{\varphi} + \bar{\tilde{\boldsymbol{m}}} \cdot \delta\boldsymbol{t}\right] \, \mathrm{d}\mathcal{A} + \int_{\partial_n\mathcal{A}} \bar{\tilde{\boldsymbol{n}}} \cdot \delta\boldsymbol{\varphi} \, \mathrm{d}s + \int_{\partial_m\mathcal{A}} \bar{\tilde{\boldsymbol{m}}} \cdot \delta\boldsymbol{t} \, \mathrm{d}s \tag{3.4}$$

where the prescribed distributed force on the boundary $\partial_n\mathcal{A}$ is $\bar{\tilde{\boldsymbol{n}}} = \boldsymbol{n}^\alpha\nu_\alpha$ and the prescribed torque on the boundary $\partial_m\mathcal{A}$ is $\bar{\tilde{\boldsymbol{m}}} = \tilde{\boldsymbol{m}}^\alpha\nu_\alpha$. The one-form normal to the boundary (and lying in the tangent plane to the surface) is denoted as $\boldsymbol{\nu} = \nu_\alpha\boldsymbol{\varphi}^\alpha$.

3.1 Strain measures

The displacement vector is introduced as $\boldsymbol{u} = \boldsymbol{\varphi} - \boldsymbol{\varphi}^0$. The linear membrane and bending strain measures can be derived from the kinematic variables in (3.1) and (3.2) in the form

$$\varepsilon_{\alpha\beta} = \tfrac{1}{2}(\boldsymbol{\varphi}^0_{,\alpha} \cdot \boldsymbol{u}_{,\beta} + \boldsymbol{\varphi}^0_{,\beta} \cdot \boldsymbol{u}_{,\alpha}) \; , \tag{3.5}$$

$$\rho_{(\alpha\beta)} = \tfrac{1}{2}(\boldsymbol{\varphi}^0_{,\alpha} \cdot \Delta\boldsymbol{t}_{,\beta} + \boldsymbol{\varphi}^0_{,\beta} \cdot \Delta\boldsymbol{t}_{,\alpha} + \boldsymbol{u}_{,\alpha} \cdot \boldsymbol{t}^0{}_{,\beta} + \boldsymbol{u}_{,\beta} \cdot \boldsymbol{t}^0{}_{,\alpha}) \; . \tag{3.6}$$

where only the symmetric part of the bending strain measure has been considered, as demonstrated by the indices being enclosed in parentheses.

The Kirchhoff-Love hypothesis needs to be finally introduced explicitly to obtain the definite forms for the strain measures. The mathematical form of this hypothesis reads

$$\boldsymbol{t} = (\bar{j})^{-1} \left(\boldsymbol{\varphi}_{,1} \times \boldsymbol{\varphi}_{,2}\right) \; , \quad \|\boldsymbol{t}\| = 1 \; . \tag{3.7}$$

The implications are, that we can write derivatives and increments of the director in terms of the covariant basis vectors $\boldsymbol{\varphi}_{,\alpha}$. Straightforward manipulation gives for the derivatives of the normal vector in the reference configuration $\boldsymbol{t}^0$

$$\boldsymbol{t}^0{}_{,\alpha} = (\bar{j}^0)^{-1} \left(\boldsymbol{\varphi}^0_{,1\alpha} \times \boldsymbol{\varphi}^0_{,2} + \boldsymbol{\varphi}^0_{,1} \times \boldsymbol{\varphi}^0_{,2\alpha}\right) \tag{3.8}$$

Next, the linear part of the increment $\Delta\boldsymbol{t} = \boldsymbol{t} - \boldsymbol{t}^0$ will be derived. Using the definition (3.7) of the normal vector, the following relation can be derived by retaining only terms linear in $\boldsymbol{u}$, and by invoking the condition $\boldsymbol{t}_{,\alpha} \cdot \boldsymbol{t} = 0$ which can be obtained by differentiation of the relation $\|\boldsymbol{t}\| = 1$

$$\Delta\boldsymbol{t} = \boldsymbol{t} - \boldsymbol{t}^0 \approx (\bar{j}^0)^{-1} \left(\boldsymbol{u}_{,1} \times \boldsymbol{\varphi}^0_{,2} + \boldsymbol{\varphi}^0_{,1} \times \boldsymbol{u}_{,2} \right) \ . \tag{3.9}$$

Similarly to the derivation of equation (3.8), we can obtain the relations for partial derivatives of the increment $\Delta\boldsymbol{t}$:

$$\Delta\boldsymbol{t}_{,\alpha} = (\bar{j}^0)^{-1} \left(\boldsymbol{u}_{,1\alpha} \times \boldsymbol{\varphi}^0_{,2} + \boldsymbol{u}_{,1} \times \boldsymbol{\varphi}^0_{,2\alpha} + \boldsymbol{\varphi}^0_{,1\alpha} \times \boldsymbol{u}_{,2} + \boldsymbol{\varphi}^0_{,1} \times \boldsymbol{u}_{,2\alpha} \right) \tag{3.10}$$

The membrane strain measures of equation (3.5) are not affected by the introduction of the Kirchhoff-Love hypothesis. On the other hand the bending strain measures can be rewritten as

$$\begin{aligned}
\rho_{11} &= -\boldsymbol{u}_{,11} \cdot \boldsymbol{t}^0 + (\bar{j}^0)^{-1} \left[\boldsymbol{u}_{,1} \cdot \left(\boldsymbol{\varphi}^0_{,11} \times \boldsymbol{\varphi}^0_{,2} \right) + \boldsymbol{u}_{,2} \cdot \left(\boldsymbol{\varphi}^0_{,1} \times \boldsymbol{\varphi}^0_{,11} \right) \right] \ , \\
\rho_{22} &= -\boldsymbol{u}_{,22} \cdot \boldsymbol{t}^0 + (\bar{j}^0)^{-1} \left[\boldsymbol{u}_{,1} \cdot \left(\boldsymbol{\varphi}^0_{,22} \times \boldsymbol{\varphi}^0_{,2} \right) + \boldsymbol{u}_{,2} \cdot \left(\boldsymbol{\varphi}^0_{,1} \times \boldsymbol{\varphi}^0_{,22} \right) \right] \ , \\
\rho_{12} &= -\frac{1}{2} \left(\boldsymbol{u}_{,12} + \boldsymbol{u}_{,21} \right) \cdot \boldsymbol{t}^0 \\
&\quad + \frac{1}{2} (\bar{j}^0)^{-1} \left[\boldsymbol{u}_{,1} \cdot \left(\left(\boldsymbol{\varphi}^0_{,12} + \boldsymbol{\varphi}^0_{,21} \right) \times \boldsymbol{\varphi}^0_{,2} \right) + \boldsymbol{u}_{,2} \cdot \left(\boldsymbol{\varphi}^0_{,1} \times \left(\boldsymbol{\varphi}^0_{,12} + \boldsymbol{\varphi}^0_{,21} \right) \right) \right] \ .
\end{aligned} \tag{3.11}$$

Using the symmetry with respect to partial differentiation, $\boldsymbol{\varphi}^0_{,12} = \boldsymbol{\varphi}^0_{,21}$ and $\boldsymbol{u}_{,12} = \boldsymbol{u}_{,21}$, the third equation of (3.11) can be simplified to

$$\rho_{12} = -\boldsymbol{u}_{,12} \cdot \boldsymbol{t}^0 + (\bar{j}^0)^{-1} \left[\boldsymbol{u}_{,1} \cdot \left(\boldsymbol{\varphi}^0_{,12} \times \boldsymbol{\varphi}^0_{,2} \right) + \boldsymbol{u}_{,2} \cdot \left(\boldsymbol{\varphi}^0_{,1} \times \boldsymbol{\varphi}^0_{,12} \right) \right] \ . \tag{3.12}$$

3.2 Constitutive equations

Let us consider only the simplest form of constitutive equations, namely the isotropic elasticity. If both the effective stress resultants and the linearized strain measures are arranged in vectors, we can write the isotropic hyperelasticity in matrix form

$$\left\{ \begin{array}{c} \tilde{n}^{11} \\ \tilde{n}^{22} \\ \tilde{n}^{12} \end{array} \right\} = \frac{Eh}{1-\nu^2} \boldsymbol{C} \left\{ \begin{array}{c} \varepsilon^{11} \\ \varepsilon^{22} \\ 2\varepsilon^{12} \end{array} \right\} \ , \quad \left\{ \begin{array}{c} \tilde{m}^{11} \\ \tilde{m}^{22} \\ \tilde{m}^{12} \end{array} \right\} = \frac{Eh^3}{12(1-\nu^2)} \boldsymbol{C} \left\{ \begin{array}{c} \rho^{11} \\ \rho^{22} \\ 2\rho^{12} \end{array} \right\} \ , \tag{3.13}$$

where the matrix $\boldsymbol{C}$ is given as

$$\boldsymbol{C} = \left[\begin{array}{ccc} (a^{011})^2 & \left(\begin{array}{c} \nu a^{011} a^{022} + \\ (1-\nu)(a^{012})^2 \end{array} \right) & a^{011} a^{012} \\ & (a^{022})^2 & a^{022} a^{012} \\ \text{symm} & & \frac{1}{2} \left(\begin{array}{c} (1-\nu) a^{011} a^{022} \\ +(1+\nu)(a^{012})^2 \end{array} \right) \end{array} \right] \tag{3.14}$$

with $a^{0\alpha\beta}$ as the components of the first fundamental form in the dual basis.

4 Moving Least Squares technique

The Element-Free Galerkin method uses the moving least-squares approximation (MLS) to construct the numerical discretization and also the surface shape approximation. The MLS have been used in statistics under the name of "loess" (local regression) to fit curves or surfaces to scattered data since the 1920's – *cf.* details in [7] and references therein.

The starting point of the Element-Free Galerkin method (EFGM) is the following equation, which approximates a function $u(\boldsymbol{x})$ in a small neighbourhood of $\boldsymbol{x}$ by a (seemingly) polynomial expansion (actually, the approximation is much more complicated; for instance, it is rational when a a polynomial weight function is used):

$$u(\boldsymbol{x}) = p_j(\boldsymbol{x}) a_j(\boldsymbol{x}) \ , \quad j = 1, \ldots, n \tag{4.1}$$

The polynomial basis $p_j(\boldsymbol{x})$ is known, the unknown coefficients $a_j(\boldsymbol{x})$ are solved for by the moving least-squares procedure using prescribed values u_I at nodal points $\boldsymbol{x}_I$, $I = 1, \ldots, M$. As is well known, the approximation (4.1) *must be at least quadratic* when applied to fourth-order problems (see e.g. Strang and Fix' book [23]). The reason is, that the governing weak form contains second-order derivatives, so that a quadratic polynomial must be represented exactly by (4.1), for the purpose of consistency. Although equation (4.1) is in general of degree higher than that of $p_j(\boldsymbol{x})$, the above requirement should hold for the choice $a_j(\boldsymbol{x}) = const.$ Consequently, the MLS approximation with a quadratic basis will represent a quadratic polynomial exactly. The polynomial basis adopted in this work was (i) quadratic, ie. the "minimal" basis, and (ii) quartic, which was used to remove membrane locking as discussed in section 13:

$$\begin{aligned} &\text{(i)} \quad \{p_j(\boldsymbol{x})\}^T = \left\{1, x, y, x^2, xy, y^2\right\}^T , \ (n = 6), \\ &\text{(ii)} \quad \{p_j(\boldsymbol{x})\}^T = \left\{1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3, x^4, x^3y, x^2y^2, xy^3, y^4\right\}^T , \ (n = 15). \end{aligned} \tag{4.2}$$

Note, that for the actual calculations the argument $\boldsymbol{x}$ should be replaced by simple linear transformation $\bar{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}_{orig}$ to shift the origin to the evaluation point. Otherwise, a loss of accuracy follows from the absolute values of $\boldsymbol{x}$ being too large.

The moving least-squares technique consists in minimizing the weighted L_2 norm

$$J = \sum_{I=1}^{M} w(\boldsymbol{x} - \boldsymbol{x}_I) \left[p_j(\boldsymbol{x}_I) a_j(\boldsymbol{x}) - u_I\right]^2 \ , \tag{4.3}$$

where $w(\boldsymbol{x} - \boldsymbol{x}_i)$ is a weight function of compact support (often called the domain of influence of node i).

This yields the following linear system of equations for the coefficients a_j:

$$\boldsymbol{A}(\boldsymbol{x}) \{a_j\} = \boldsymbol{B}(\boldsymbol{x}) \{u_m\} \ , \qquad \{a_j\} \in R^n, \{u_m\} \in R^M \tag{4.4}$$

where M is the number of EFG nodes whose domain of influence includes $\boldsymbol{x}$, and

$$[\boldsymbol{A}(\boldsymbol{x})]_{ij} = \sum_{m=1}^{M} w(\boldsymbol{x}-\boldsymbol{x}_m) p_i(\boldsymbol{x}_m) p_j(\boldsymbol{x}_m) \ ,$$

$$\boldsymbol{B}(\boldsymbol{x}) = \left[w(\boldsymbol{x}-\boldsymbol{x}_1) \left\{ p_i(\boldsymbol{x}_1) \right\}, \ldots, w(\boldsymbol{x}-\boldsymbol{x}_M) \left\{ p_i(\boldsymbol{x}_M) \right\} \right] \ .$$

The equation (4.1) can thus be put into standard form

$$u(\boldsymbol{x}) = \left\{ \phi_I(\boldsymbol{x}) \right\}^T \left\{ u_I \right\} \ , \qquad \left\{ \phi_I \right\} \in R^M, \left\{ u_I \right\} \in R^M \ , \tag{4.5}$$

with $\phi_I(\boldsymbol{x})$ being the *shape functions*

$$\phi_I(\boldsymbol{x}) = p_j \left[\boldsymbol{A}(\boldsymbol{x})^{-1} \cdot \boldsymbol{B}(\boldsymbol{x}) \right]_{jI} \ . \tag{4.6}$$

The continuity of the shape function $\phi_I(\boldsymbol{x})$ is governed by the continuity of the basis functions p_j, and by the smoothness of the matrices $\boldsymbol{A}(\boldsymbol{x})^{-1}$ and $\boldsymbol{B}(\boldsymbol{x})$. The latter is governed by the smoothness of the weight function.

5 Description of the shell surface

The present methodology is targeted at general shells. It means that we have to deal with the issue of surface shape approximation. The shapes that we need to consider here are free-form surfaces, as only these are general enough. To approximate the shape of the shell the moving-least squares technique is applied to fit the approximate surface to a collection of scattered data points. It should be noted that the requirements posed on the approximate surface are governed here by the desired mechanical properties of the surface. These questions will addressed subsequently.

5.1 Surface approximation

The moving least squares technique can be applied immediately to obtain the surface approximation. Let us assume that a set of M (scattered) points in space is given. These points lie directly on the surface to be approximated at locations $\boldsymbol{x}_I$. One way of obtaining these points is to use vertices of a finite element mesh (or of another polygonal tesselation of the surface as it is sometimes produced by geometric modelers for different purposes such as visualization). The approximate surface then may be described by

$$\boldsymbol{\varphi}(\boldsymbol{\xi}) = \left\{ p_j(\boldsymbol{\xi}) \right\}^T \left\{ \boldsymbol{a}_j(\boldsymbol{\xi}) \right\} = \phi_I(\boldsymbol{\xi}) \boldsymbol{x}_I \ , \tag{5.1}$$

where $\boldsymbol{\xi}$ is the parameterization of the surface. Note, that the unknown coefficients are now 3D position vectors. Note also, that the surface constructed does not in general pass through the prescribed points, ie. the technique is *not interpolating*.

REMARK 1: The use of a finite element mesh for the purpose of shape definition/ numerical integration seems to be of value also with respect to coupling with the finite element technique – *cf.* Belytschko *et al.* [3]. In that case, the finite element mesh would be readily available without additional cost.

REMARK 2: It should be noted that the continuity of the surface aproximation was discussed in general terms in section 4. As at least C^1 continuity is required for shape functions in shells (and to satisfy no-strain rigid body motion conditions, the same shape functions should be used both for shape and displacement approximation), special techniques must be adopted to model surfaces with creases and similar discontinuities in slope, e.g. such surfaces might have to be split with appropriate boundary conditions imposed at the seam.

6 Quality of surface shape approximation

The approximate geometry of equation (5.1) must be evaluated with respect to several approximation quality criteria, before it can be used in mechanical computations. Let us summarize the aspects of interest in the computations:

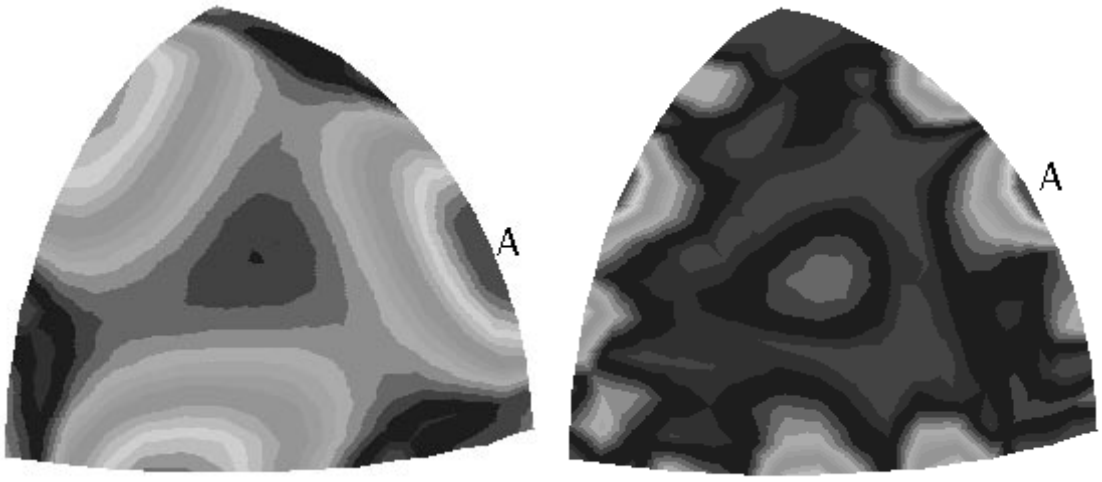
- Quality of surface shape description.
- Quality of boundary conditions description.
- Properties of the EFG approximation with respect to stiffness matrix evaluation.

REMARK: It should be noted, that the input data to the program working with the EFG approximation will not in general include sufficient geometric information about the exact surface shape. The minimal input that can be expected is a finite element discretization of the exact surface. Therefore, the task of shell surface approximation needs to be reformulated as "approximation of the approximate" shape. A general procedure how to construct an approximation of the (approximate) shape was therefore developed and is described in the following.

6.1 Overall shape similarity

The surface shape should be approximated sufficiently closely, so that its mechanical properties, which are governed in most cases by its curvatures and by the "smoothness" of the shape, are sufficiently close to properties of the exact surface representation.

The approximate surface should be smooth (with an intuitive definition of "smooth"). In case the approximate shape shows "bumps" or "dimples" they stiffen the shell in bending, and a significant part of strain energy might get lost in membrane action. There is a need for systematic assessment of the smoothness approximation accuracy, which has not yet been addressed by the authors.



(a) quadratic basis ($\mathcal{E}_{max} = 2.75 \times 10^{-3} R$) (b) quartic basis ($\mathcal{E}_{max} = 3.3 \times 10^{-4} R$)

Figure 1: Color-encoded maps of error in distance of the points of the approximate surface from the center with respect to the exact sphere (largest error $\mathcal{E}_{max}$ at point "A").

Example 1: The approximation accuracy of the present scheme can be shown on the example of a segment of a spherical surface. The difference between the exact and approximate shapes is measured as error in distance from the center of the points on the approximate surface with respect to the exact radius of the sphere. The results are plotted as color-encoded maps in figure 1, where is shown (a) the accuracy of the scheme (5.1) using quadratic basis (maximum error $\mathcal{E}_{max} = 2.75 \times 10^{-3} R$ at point "A"), and (b) the accuracy of (5.1) with quartic basis (maximum error $\mathcal{E}_{max} = 3.3 \times 10^{-4} R$ at point "A"). The parametric space of the surface is shown in figure 2. A circle centered at "C" has been added to show good symmetry properties of the parametric space. Note, that despite the fact that the non-symmetry does not show in figure 2, it can be clearly seen in figure 1.

6.2 Preservation of symmetry

Further, the shape approximation should preserve the symmetry of the structure. Problems with symmetry preservation are not limited to EFG – it is also difficult for finite element meshes (especially those generated automatically). Examples are

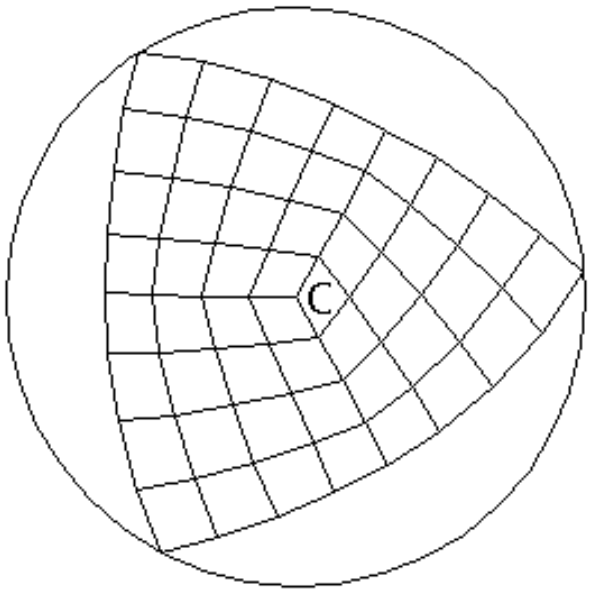


Figure 2: The parametric space of the spherical surface.

obvious: approximations to cylinders should have the axis of the cylinder as one of the symmetry axes, symmetric structure with symmetric boundary conditions should also be symmetric after shape approximation. The present scheme cannot guarantee that the approximate surface will preserve planes of symmetry. However, there are devices to ameliorate the discrepancies, as discussed below.

6.3 Developability

Another issue is the preservation of zero Gaussian curvature: Approximations of developable surfaces should preserve the developability. Otherwise artificial stiffenening results, which makes the shell response too stiff. The present scheme can produce developable surface from a developable background mesh, if the vertices of the mesh are converted into EFG nodes. However, introduction of additional EFG nodes must be done in such a manner so as not to change the shape of the approximate surface. It means that their locations would have to be computed from appropriate constraints.

6.4 Evaluation of integrals

The mechanical properties of the shell are represented by the global stiffness matrix. This stiffness matrix is evaluated by numerical integration over the surface area. Therefore, the area of the approximate surface should be close to the area of the original surface. The scheme for surface approximation, which is described below, seems to yield rather good results in this respect. Another aspect of the numerical integration is the error introduced by the finite number of evaluation points. It is clear, that the error depends on the smoothness of the evaluated function. Therefore, if the

Jacobian of the mapping from the parametric space to the surface is rapidly varying, the error incurred by numerical quadrature grows. A more systematic approach to the assessment of this aspect is needed.

6.5 Curvature approximation

The mechanical properties of the shell depend crucially on the second fundamental form of the surface, i.e. on its curvatures. The approximation of curvatures was not yet studied in detail, and more systematic approach is needed, for instance one of those discussed in computer graphics literature.

6.6 Boundary approximation

The mechanical response of a shell is in many cases affected by its boundary conditions. Especially accuracy of *essential boundary conditions* approximation seems to be important. Let us consider the effect of the moving least-squares approximation on the shape of the shell surface. The shape depends in general on the size of the support of the weight functions $w(\boldsymbol{x} - \boldsymbol{x}_I)$. The larger the support, the closer is the moving least-squares procedure to the ordinary least-squares. If the polynomial basis p_i is quadratic, and if the support of the weight functions is very large, the shape approximation becomes essentially a trimmed piece of quadric, independently of the actual curvature of the original shell surface. It means that the approximate surface is always flatter than the exact one, if the points $\boldsymbol{x}_I$ lie on the exact surface. Consequently, the normals to the approximate surface are not identical to the exact normals. If an essential boundary condition depends on the direction of the support, as it is the case for instance in sliding support restricting rotation about the tangent to the boundary curve, the error introduced into the modelling of the mechanical properties becomes significant (*cf.* figure 3). The same holds in other cases of mechanical supports.

Another aspect is the planarity of boundary curves. If a boundary curve should be planar, as for example in planar cuts through a shell, or on the plane of symmetry, then non-planarity may cause artificial stiffening. Assume, for instance, that an edge of the shell surface becomes wavy after EFG approximation. No-energy sliding in the tangential direction then becomes impossible, as membrane stresses are generated (see figure 3). The result is a correct answer to a wrong question.

7 Parameterization

In order to be able to use the equation (5.1) in the context of moving least squares, the surface must be *parameterized*, ie. the parametric space $\mathcal{A}$ of equation (2.1) must be defined. The approach used in the present work is based on the fact, that a

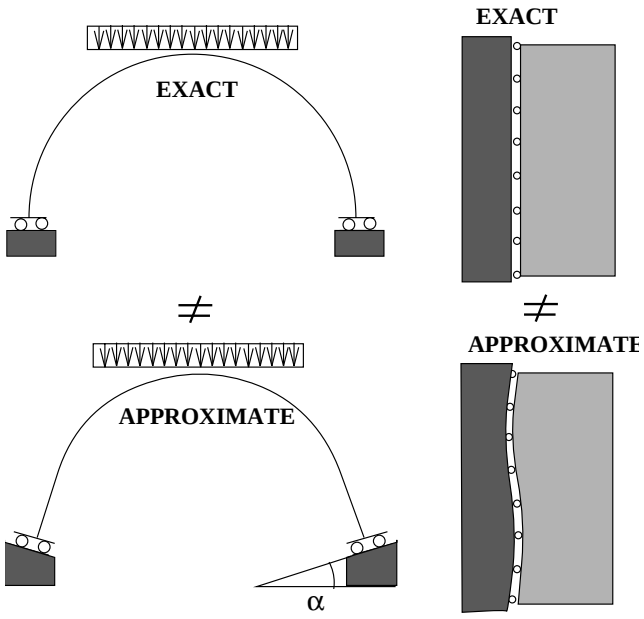


Figure 3: Clamped conditions for an arch and sliding boundary condition for an edge of a surface. The mechanical effects are different for approximate boundary conditions.

polygonalization of the surface at hand is usually readily available (e.g. in the form of a finite element mesh). It can describe a surface of arbitrary complexity – of irregular shape (such as trimmed patches), surfaces with self-intersecting boundary, with any number of holes, and both closed and open surfaces. Therefore, it is possible to use FE mesh nodes to make them into EFG nodes $\boldsymbol{x}_I$, and to define the approximate surface shape as $\boldsymbol{\varphi}(\boldsymbol{\xi}) = \phi_I(\boldsymbol{\xi})\boldsymbol{x}_I$, where the parameterization $\boldsymbol{\xi}$ is defined by standard FE interpolation on the FE mesh $\boldsymbol{\xi} = N_K\boldsymbol{\xi}_K$, where $\boldsymbol{\xi}_K$ are the nodal values of the parameter $\boldsymbol{\xi}$.

REMARK: Note, that C^0 continuity of the polygonal tesselation is *not* required in the present approach, ie. the polygonalization may be incompatible.

The way in which the values of the parameter $\boldsymbol{\xi}_K$ are assigned to the nodes is crucial to the shape approximation. The smoothness of the mapping from the parametric to the physical space appears in the integration formulas through the area measure $\mathrm{d}\mathcal{A} = \bar{j}\mathrm{d}\xi^1\mathrm{d}\xi^2$. Ideally, the Jacobian $\bar{j}$ should be constant, e.g. $\bar{j} \approx 1$. Therefore, the parameterization should be such as to yield the Jacobian $\bar{j}$ as close to constant value as possible.

A good choice seem to be the intrinsic coordinates. They are defined for a number of simple surfaces (e.g. quadrics), but they are difficult to define for general (free-form) surfaces. Therefore, to accomodate general shapes, an algorithm for generating the nodal parameter values $\boldsymbol{\xi}_K$ automatically has been developed.

7.1 Parameterization algorithm

Surface fitting techniques usually work with the real plane as the parametric space. The surfaces considered here are on the contrary trimmed. If we define the parameterization on the background mesh, the resulting surface approximation becomes in fact a *trimmed moving least squares patch*. The usual definition of a patch can be used to define the surfaces the present algorithm can handle. The patch should have (in the parametric space!):

- An exterior boundary described by a single, non-self-intersecting curve of arbitrary shape.
- Any number of interior boundaries (holes) of arbitrary shape. Also these curves must be non-self-intersecting.

Consider as an example of a surface that cannot be handled by the algorithm a cylinder of figure 4. The points on the straight line passing through the point shown as a large black dot cannot be assigned unambiguous values of the parameters $\boldsymbol{\xi}$. This is demonstrated by the circular path with an arrow. The intrinsic Gaussian coordinates are no longer single-valued. To avoid these complexities at this stage, only the simple case specified above is considered in this work. However, the need to represent more general surfaces can be covered easily by splitting the original surface into pieces that can be represented as patches, and by glueing them together by continuity boundary conditions, which are easily accomodated by the Lagrange multiplier technique.

7.1.1 General remarks

The algorithm will be explained on C-language fragments. These were slightly edited to provide the desired effect with the least fuss and bother concerning syntax completeness, naming conventions etc. The advantage of this approach is that the description is concise.

The crucial data structure used in the algorithm is a *queue*. The behaviour of a queue is best described by the operations that can be performed on it. The function names **make_queue**() and **destroy_queue**(q) are self-explanatory. Function **enqueue**(q, bme) adds an object bme to the queue in such a manner, that it becomes the "last" object currently in the queue. Function **dequeue**(q) takes the "first" object in the queue out (so that it is no longer present in the queue when the operation finishes). The second object in the queue thus becomes the first one, and so on. Function **queue_empty**(q) enquires whether the queue contains any elements.

The abbreviation BME stands for "background mesh element". BME is a generic element – it could be quadrilateral or triangular, with 4 or 9 nodes, etc. The procedures accepting BME are therefore polymorphic (the appropriate code is chosen at run-time).

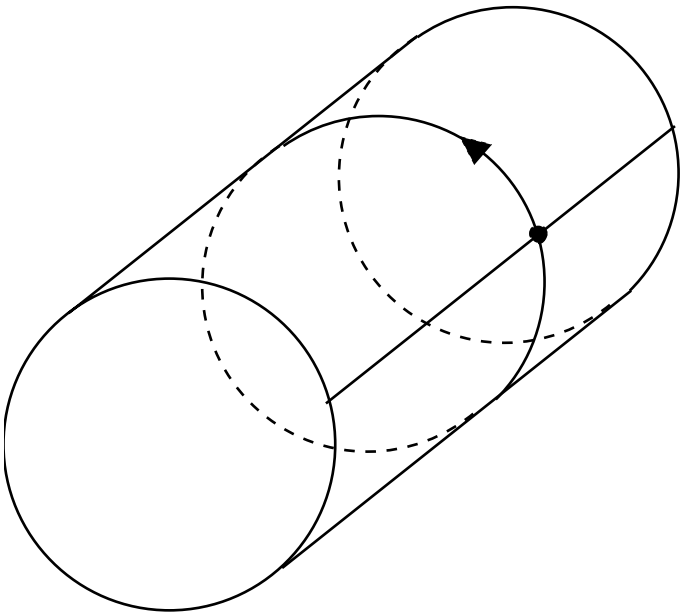


Figure 4: Cylindrical shell surface demonstrating difficulties in the definition of unambiguous parameterization.

7.1.2 Description of the algorithm

Procedure **parameterize**()

This is the top-level procedure. It generates the parameterization of a single surface patch. The Boolean variable p_started is used to flag whether the parameterization has been started.

Procedure **select_start_bme()**

The algorithm rolls off by selecting the first BME to be placed in the queue. This decision has major impact on the results, as (i) it affects the way in which the normals to the surface are computed, and (ii) it affects the parameterization quality for non-developable surfaces. The point (ii) will be explained next. It will be assumed here, that the starting BME was provided as input data (i.e. it was selected by the user). The normal of the first BME is fixed inside **select_start_bme**(), so that the first BME can be considered "homogenized" (see below).

The "while" loop is then entered. An BME is taken from the queue, and its input data are first adjusted so that the normal is computed correctly for all BME's on the surface patch.

Procedure **homogenize_normal**(*bme*)

Correctly means that the normals point in the "same" direction when the surface is viewed from one of its sides (with the obvious, intuitive, definition of the "same

```
GLOBAL Boolean p_started = False;/* not yet */
PROC
parameterize(Surface *s)
{
    BkgMeshElem *bme;
    BMEQueue queue = make_queue(); /* The integration-unit queue */
    bme = select_start_bme(s); /* choose the first bme */
    enqueue(queue, bme); /* push into queue */
    p_started = False;/* not yet */
    while (!queue_empty(queue)) {
        bme = dequeue(queue); /* pop the queue */
        if (homogenize_normal(bme)) {
            parameterize_bme(bme, queue);
        } else {
            enqueue(queue, bme);/* Failed: try later on the same bme */
        }
        p_started = True;
    }
    destroy_queue(queue);
}
```

Figure 5: Procedure **parameterize**().

direction"). This operation is called *normal homogenization*. The function does nothing, if the BME normal has been homogenized before, or if it has been preset by the input data; otherwise returns "True" if the normal of the BME has been homogenized successfully; "False" otherwise. To homogenize the normal, the BME queries whether any of its neighbouring BME's has been homogenized (i.e. whether its normal can be considered for comparison). The input data (in most cases this means the order of the vertices) is then adjusted, so that the normal of the BME handled agrees with the normal of the neighbour.

Procedure **parameterize_bme**()

Once the normal has been homogenized, the parameter values can be generated at the vertices of the BME. There are two cases here: (i) parameterization of the first BME on the surface, and (ii) parameterization of any other BME.

Let us first consider case (i): A local cartesian two-dimensional system $\langle \hat{x}; \hat{y} \rangle$ is constructed in the plane of the BME in an arbitrary way. The coordinates $\langle \hat{x}_K; \hat{y}_K \rangle$ of the vertex K are taken as the surface parameters: $\xi^1_K = \hat{x}_K$, $\xi^2_K = \hat{y}_K$.

Let us now consider the case (ii), with two subcases: (iia) Planar surface (i.e. the shell is actually a plate), and (iib) general surface. Considering first the case (iia), it should be noted that the coordinate system constructed above for the first BME, can be used analogously for all subsequent BME's. This resolves the case.

The case (iib) of general surfaces is much more difficult, and a completely satisfactory solution is yet to be found. The following scheme works quite well for developable surfaces on which a developable finite element mesh has been constructed (e.g. a mesh on a cylinder following the straight generatrix), and also for flat non-developable surfaces with "nice" finite element covering (nice means almost regular mesh). The reason will be given in what follows.

The functionality of **parameterize_bme**() can be described in general terms (using psedo-code) as shown in figure 7.1.2. The action **1.** is self-explanatory. The action **2.** has been described above as the case (i). The actions **3.** and **6.** place into the queue all BME's which share an edge with the currently handled BME. The net effect of these two actions is a "flooding" effect – the parameterized BME's spread over the surface as if it was flooded.

The action **5.** is the most tricky part of the algorithm. It depends on the type of the BME involved (triangle, quadrangle, etc.). Let us describe the action for a three-noded triangle first: Two of the triangles vertices may have been assigned the parameters before. In that case we can compute the parameters at the third vertex by noting that coordinates of two vertices in an arbitrary cartesian system determine the coordinates of third one. (Actually, the problem is overdetermined by four values, so the length of the triangle side with the two vertices given must be scaled.)

Analogously can be computed the parameters at two remaining vertices in a quadrilateral, when the vertices on any side have been assigned parameters previously. The task to compute the coordinates of the fourth vertex in a quadrilateral

```
PROC
parameterize_bme(BkgMeshElem *bme, BMEQueue queue)
{
    ⟨ 1. denote vertices of the BME that were assigned parameterization
        before as "bound" vertices; denote the rest as "free" vertices ⟩
    if (number of "bound" vertices EQUAL 0) {
        ⟨ 2. assign initial parameterization ⟩
        ⟨ 3. push neighbours to queue ⟩
    } else if (number of "bound" vertices EQUAL 1) {
        ⟨ 4. push BME back to queue: we don't know which direction to go ⟩
    } else { /* the general case */
        ⟨ 5. compute parameterization of the "free" vertices ⟩
        ⟨ 6. push neighbours to queue ⟩
    }
}
```

Figure 6: The function **parameterize_bme**() in pseudo-code.

(with three vertices having been assigned parameters) can be performed by first resolving the clash due to overdeterminacy (the problem requires setting the third vertex of two triangles, with the constraint of computing the same location by using both of them; the solution is not unique.)

Let us note that the overdeterminacy of the problems above is due to the *angular defect*, associated to the Gaussian curvature of the surface (see Calladine in [6]).

The action **4.** is a fall-back for the case that only one vertex has been assigned parameter values before. In that case, the BME cannot be parameterized and its processing is postponed.

Now it becomes clear, why the choice of the first BME was important. For the non-developable surfaces the present algorithm will accumulate errors as it proceeds with the parameterization by flooding. Therefore, it is advantageous to start from the geometrical center of the surface to minimize the distance the algorithm has to travel to the edges.

8 Displacement parameters

There are at least two ways how to write the approximation of displacements. First, it is possible to write the displacements in the curvilinear basis $\boldsymbol{\varphi}^0_{,\alpha}$.

$$\boldsymbol{u}(\boldsymbol{\xi}) = \sum_{I=1}^{M} \phi_I(\boldsymbol{\xi}) \left[U_I \boldsymbol{\varphi}^0_{,1}(\boldsymbol{\xi}) + V_I \boldsymbol{\varphi}^0_{,2}(\boldsymbol{\xi}) + W_I \boldsymbol{t}^0(\boldsymbol{\xi}) \right] \ . \tag{8.1}$$

However, this approach has a serious drawback: The covariant derivatives then become indispensable. This means that the equations become rather complicated

(the derivatives of the basis vectors have to be evaluated), and what even more impairs the applicability of this approach is, that third order derivatives of the shape functions must be computed. These are (i) unreliable (the higher the derivative, the less accuracy in the approximation), and (ii) expensive to compute (look at the equation (4.6) – the number of matrix terms grows quickly with each differentiation).

The other approach is very simple. Just express the displacement vector in the global cartesian basis $\boldsymbol{E}_k$.

$$\boldsymbol{u}(\boldsymbol{\xi}) = \sum_{I=1}^{M} \phi_I(\boldsymbol{\xi}) \left[U_I \boldsymbol{E}_1 + V_I \boldsymbol{E}_2 + W_I \boldsymbol{E}_3\right] = \sum_{I=1}^{M} \phi_I(\boldsymbol{\xi}) \boldsymbol{\mathcal{U}}_I \ . \tag{8.2}$$

REMARK 1. The approximation introduced in (8.2) is homogeneous in the polynomial order for all components of the displacement vector. This spells problems with membrane locking. The causes of membrane (and shear) locking were studied by Belytschko et al. in [4], and by Stolarski et al. in [22]. The present EFG approximation has a very interesting property with respect to membrane locking in that it can be alleviated without resorting to projections or underintegration. More on this subject is presented below.

REMARK 2. Note, that to evaluate the derivatives of the displacement vector $\boldsymbol{u}(\boldsymbol{\xi})$ of (8.2) one needs only to differentiate the shape function $\phi_I(\boldsymbol{\xi})$. This means that the smoothness of the displacement approximation depends on the smoothness of the shape function. To compute the strains one needs only second order derivatives with respect to the parameters $\boldsymbol{\xi}$ (recall the equation (3.11)).

9 Stiffness matrix

The stiffness matrix can be evaluated by following standard Galerkin procedure, substituting results of preceding developments. The crucial issue is the construction of the strain-displacement matrices. These will be detailed below. The stiffness matrix (and for that matter all matrices) is evaluated by numerical integration. Usual Gaussian integration has been used throughout.

It should be noted, that one of the characteristic features of the EFG method is the variable number of nodal points influencing an integration point. Therefore, it is convenient to write the strain-displacement matrices in a symbolic form as a sum of submatrices (one for each nodal point involved, i.e. such a node, whose nodal parameters have support interfering with the integration point in question) rather than a matrix with variable dimension. Note, that summation over repeated index is implied in some equations.

9.1 Membrane strain-displacement matrix

The membrane strain-displacement matrix $\boldsymbol{B}_{(m)_I}$ for the Ith nodal point is obtained by substituting approximation (8.2) into (3.5) and gathering the membrane strains

$\varepsilon^{\alpha\beta}$ in the vector of equation (3.13).

$$\left\{ \begin{array}{c} \varepsilon^{11} \\ \varepsilon^{22} \\ 2\varepsilon^{12} \end{array} \right\} = \left[\boldsymbol{B}_{(m)_I} \right] \left\{ \begin{array}{c} U_I \\ V_I \\ W_I \end{array} \right\} = \\ \left[\begin{array}{ccc} \phi_{I,1}\boldsymbol{\varphi}^0_{,1} \cdot \boldsymbol{E}_1 & \phi_{I,1}\boldsymbol{\varphi}^0_{,1} \cdot \boldsymbol{E}_2 & \phi_{I,1}\boldsymbol{\varphi}^0_{,1} \cdot \boldsymbol{E}_3 \\ \phi_{I,2}\boldsymbol{\varphi}^0_{,2} \cdot \boldsymbol{E}_1 & \phi_{I,2}\boldsymbol{\varphi}^0_{,2} \cdot \boldsymbol{E}_2 & \phi_{I,2}\boldsymbol{\varphi}^0_{,2} \cdot \boldsymbol{E}_3 \\ \left(\begin{array}{c} \phi_{I,2}\boldsymbol{\varphi}^0_{,1} \\ +\phi_{I,1}\boldsymbol{\varphi}^0_{,2} \end{array} \right) \cdot \boldsymbol{E}_1 & \left(\begin{array}{c} \phi_{I,2}\boldsymbol{\varphi}^0_{,1} \\ +\phi_{I,1}\boldsymbol{\varphi}^0_{,2} \end{array} \right) \cdot \boldsymbol{E}_2 & \left(\begin{array}{c} \phi_{I,2}\boldsymbol{\varphi}^0_{,1} \\ +\phi_{I,1}\boldsymbol{\varphi}^0_{,2} \end{array} \right) \cdot \boldsymbol{E}_3 \end{array} \right] \left\{ \begin{array}{c} U_I \\ V_I \\ W_I \end{array} \right\} \tag{9.1}$$

9.2 Bending strain-displacement matrix

The bending strain-displacement matrix $\boldsymbol{B}_{(b)_I}$ for the Ith nodal point is obtained by substituting approximation (8.2) into (3.6) and gathering the bending strains $\rho^{\alpha\beta}$ in the vector of equation (3.13).

$$\left\{ \begin{array}{c} \rho^{11} \\ \rho^{22} \\ 2\rho^{12} \end{array} \right\} = \left[\boldsymbol{B}_{(b)_I} \right] \left\{ \begin{array}{c} U_I \\ V_I \\ W_I \end{array} \right\} \tag{9.2}$$

The individual terms of the strain-displacement matrix $\boldsymbol{B}_{(b)_I}$ are summarized here:

$$\begin{aligned} \left[\boldsymbol{B}_{(b)_I}\right]_{1m} &= (\bar{j}^0)^{-1} \left[-\bar{j}^0 \phi_{I,11} \boldsymbol{t}^0 + \phi_{I,1} \left(\boldsymbol{\varphi}^0_{,11} \times \boldsymbol{\varphi}^0_{,2} \right) + \phi_{I,2} \left(\boldsymbol{\varphi}^0_{,1} \times \boldsymbol{\varphi}^0_{,11} \right) \right] \cdot \boldsymbol{E}_m \\ \left[\boldsymbol{B}_{(b)_I}\right]_{2m} &= (\bar{j}^0)^{-1} \left[-\bar{j}^0 \phi_{I,22} \boldsymbol{t}^0 + \phi_{I,1} \left(\boldsymbol{\varphi}^0_{,22} \times \boldsymbol{\varphi}^0_{,2} \right) + \phi_{I,2} \left(\boldsymbol{\varphi}^0_{,1} \times \boldsymbol{\varphi}^0_{,22} \right) \right] \cdot \boldsymbol{E}_m \\ \left[\boldsymbol{B}_{(b)_I}\right]_{3m} &= 2(\bar{j}^0)^{-1} \left[-\bar{j}^0 \phi_{I,12} \boldsymbol{t}^0 + \phi_{I,1} \left(\boldsymbol{\varphi}^0_{,12} \times \boldsymbol{\varphi}^0_{,2} \right) + \phi_{I,2} \left(\boldsymbol{\varphi}^0_{,1} \times \boldsymbol{\varphi}^0_{,12} \right) \right] \cdot \boldsymbol{E}_m \end{aligned} \tag{9.3}$$

10 Rigid-body motion

The rigid-body motion (RBM) should generate no strains. This requirement can be met by the EFG approximation of (8.2). It will be shown here that the requirement translates into a simple constraint on the strain-displacement matrices. This fact can be used as a help in debugging of the code.

The RBM can be in general written as

$$\boldsymbol{u}_{RB}(\boldsymbol{\xi}) = \hat{\boldsymbol{u}} + \hat{\boldsymbol{\theta}} \times (\boldsymbol{\varphi}(\boldsymbol{\xi}) - \boldsymbol{c}) \ , \tag{10.1}$$

where $\hat{\boldsymbol{u}}$ and $\hat{\boldsymbol{\theta}}$ are constant vectors, $\boldsymbol{c}$ is the center of rotation, which can be without loss of generality assumed to be $\boldsymbol{c} = \boldsymbol{0}$.

10.1 Rigid-body translation

The translation part of RBM can be written as

$$\boldsymbol{u}_{RB}(\boldsymbol{\xi}) = \hat{\boldsymbol{u}} = \phi_I(\boldsymbol{\xi})\boldsymbol{\mathcal{U}}_I = \hat{\boldsymbol{u}} \sum_{I=1}^{M} \phi_I(\boldsymbol{\xi}) = \hat{\boldsymbol{u}} \ , \tag{10.2}$$

where it was assumed that $\boldsymbol{\mathcal{U}}_I = \hat{\boldsymbol{u}}$ and the equality $\sum_{I=1}^{M} \phi_I = 1$ has been used.

The strain $\boldsymbol{\epsilon}$ (here $\boldsymbol{\epsilon}$ stands for both membrane and bending strains, and in fact for any strain) may be related to the displacement parameters by the strain-displacement matrix. The RBM does not produce any strain for RB translation if $\boldsymbol{\epsilon} = \boldsymbol{0}$, i.e. if

$$\boldsymbol{\epsilon} = \boldsymbol{B}_I \cdot \boldsymbol{\mathcal{U}}_I = \left(\sum_{I=1}^{M} \boldsymbol{B}_I \right) \cdot \hat{\boldsymbol{u}} = \boldsymbol{0} \ . \tag{10.3}$$

The equation (10.3) can be satisfied only if

$$\sum_{I=1}^{M} \boldsymbol{B}_I = \boldsymbol{0} \ . \tag{10.4}$$

10.2 Rigid-body rotation

The rotational part of RBM can be written as

$$\boldsymbol{u}_{RB}(\boldsymbol{\xi}) = \hat{\boldsymbol{\theta}} \times \boldsymbol{\varphi}(\boldsymbol{\xi}) \ , \tag{10.5}$$

where it was assumed that $\boldsymbol{c} = \boldsymbol{0}$. It can be verified, that the parameters $\boldsymbol{\mathcal{U}}_I$ describe RB rotation, if $\boldsymbol{\mathcal{U}}_I = \hat{\boldsymbol{\theta}} \times \boldsymbol{x}_I$, with $\boldsymbol{x}_I$ being the 3D location of the Ith nodal point, i.e.

$$\boldsymbol{u}_{RB}(\boldsymbol{\xi}) = \hat{\boldsymbol{\theta}} \times (\phi_I(\boldsymbol{\xi})\boldsymbol{x}_I) \ . \tag{10.6}$$

Substituting (10.6) into the strain-displacement relation, the following equation is obtained:

$$\boldsymbol{\epsilon} = \boldsymbol{B}_I \cdot \boldsymbol{\mathcal{U}}_I = \boldsymbol{B}_I \cdot \hat{\boldsymbol{\theta}} \times \boldsymbol{x}_I = \boldsymbol{B}_I \cdot \hat{\boldsymbol{\Theta}} \cdot \boldsymbol{x}_I \ , \tag{10.7}$$

where $\hat{\boldsymbol{\Theta}}$ is an antisymmetric matrix with $\hat{\boldsymbol{\theta}}$ as the axial vector. It is not probably not worth while to prove analytically that $\boldsymbol{\epsilon}$ in (10.7) is really zero. However, a numerical check was easy to implement, and it was fulfilled within machine accuracy.

11 Essential boundary conditions

An Element-Free Galerkin handling of essential boundary conditions (EBC) is awkward as the shape functions do not vanish on the boundary of the domain (*cf.* the

reference [16]). Some of the options are: (i) point collocation [8], (ii) Lagrange multipliers [2], (iii) modified Lagrange multipliers (replacement of multipliers by their physical representations in terms of reaction forces, see [16]), (iv) enforcement by finite elements at the boundary [11]. The technique (i) lacks precision, therefore it was not considered here. Approach (iii) is not attractive for Kirchhoff-Love theory, as third-order derivatives of the shape functions are needed to compute the effective forces on the boundary, and these are expensive to compute and not very accurate. The technique ad (iv) seems to be advantageous, it was not yet tested with the plates, however.

The approach selected here to enforce the essential boundary conditions is the method of Lagrange multipliers (technique ad (ii)). The disadvantages (unpleasant, but not prohibitive) are:

- Additional unknowns increase the problem size.
- Special solver is needed to handle the resulting indefinite system of linear equations with a structure that resembles that of mixed finite element methods. (Bunch-Kauffman-Parlett symmetric indefinite factorization as described in [5] was used here.)

11.1 Tangential and normal displacements

The translational displacement conditions on a boundary curve involve three motions:

1. motion in the direction normal to the surface (given by the vector $\boldsymbol{t}^0$),
2. motion in the tangent plane to the surface
   - 2.1 in the direction of the vector $\boldsymbol{\tau}^0$ tangent to the boundary curve,
   - 2.2 in the direction of the vector $\boldsymbol{\nu}^0$ normal to the boundary curve.

Denoting $\boldsymbol{\lambda} = \lambda_\nu \boldsymbol{\nu}^0 + \lambda_\tau \boldsymbol{\tau}^0 + \lambda_t \boldsymbol{t}^0$ the Lagrangean multipliers associated with the motion, the terms actually appearing in the augmented weak form of (3.3) are

$$
\begin{aligned}
&- \int_{\partial_m \mathcal{A}} \delta\boldsymbol{\lambda} \cdot (\boldsymbol{u} - \bar{\boldsymbol{u}}) \ \mathrm{d}s - \int_{\partial_m \mathcal{A}} \delta\boldsymbol{u} \cdot \boldsymbol{\lambda} \ \mathrm{d}s = \\
&- \int_{\partial_m \mathcal{A}} \delta\boldsymbol{\lambda} \cdot (\phi_I \boldsymbol{\mathcal{U}}_I - \bar{\boldsymbol{u}}) \ \mathrm{d}s - \int_{\partial_m \mathcal{A}} \phi_I \delta\boldsymbol{\mathcal{U}}_I \cdot \boldsymbol{\lambda} \ \mathrm{d}s \ . \qquad (11.1)
\end{aligned}
$$

These integrals can be easily evaluated by approximating variation of the Lagrangean multipliers along the boundary curve by the usual form

$$
\boldsymbol{\lambda}(s) = N_K(s) \boldsymbol{\lambda}_K \ , \qquad (11.2)
$$

where N_K are standard Lagrangean shape functions. The linear shape functions with one-point quadrature were used in all examples described in the subsequent chapters.

11.2 Rotations about tangent to the boundary

The small rotation vector can be expressed from the obvious approximation of the increment of the normal director

$$\Delta \boldsymbol{t} = \boldsymbol{t} - \boldsymbol{t}^0 \approx \boldsymbol{\theta} \times \boldsymbol{t}^0 \ , \tag{11.3}$$

where $\boldsymbol{\theta}$ is the small-rotation vector. Using results of equation (3.9), the rotation vector $\boldsymbol{\theta}$ can be written as

$$\boldsymbol{\theta} = (\bar{j}^0)^{-1} \left[(\boldsymbol{u}_{,2} \cdot \boldsymbol{t}^0)\boldsymbol{\varphi}^0_{,1} - (\boldsymbol{u}_{,1} \cdot \boldsymbol{t}^0)\boldsymbol{\varphi}^0_{,2} \right] \ . \tag{11.4}$$

The component θ_τ of the rotation about the tangent vector $\boldsymbol{\tau}^0$ is obtained readily as $\theta_\tau = \boldsymbol{\theta} \cdot \boldsymbol{\tau}^0$. The terms corresponding to prescribed rotations about the tangent to the boundary, which actually appear in the augmented weak form of (3.3) are

$$- \int_{\partial_m \mathcal{A}} \delta\lambda_\tau \left(\theta_\tau - \bar{\theta}_\tau \right) \ \mathrm{d}s - \int_{\partial_m \mathcal{A}} \delta\theta_\tau \lambda_\tau \ \mathrm{d}s \tag{11.5}$$

Now, substituting for $\boldsymbol{\theta}$ from (11.4), we obtain

$$\begin{aligned} -(\bar{j}^0)^{-1} \int_{\partial_m \mathcal{A}} \delta\lambda_\tau \left[(\boldsymbol{\varphi}^0_{,1} \cdot \boldsymbol{\tau}^0)\phi_{I,2} - (\boldsymbol{\varphi}^0_{,2} \cdot \boldsymbol{\tau}^0)\phi_{I,1} \right] (\boldsymbol{\mathcal{U}}_I \cdot \boldsymbol{t}^0) \ \mathrm{d}s \\ - \int_{\partial_m \mathcal{A}} \delta\lambda_\tau \bar{\theta}_\tau \ \mathrm{d}s \\ - (\bar{j}^0)^{-1} \int_{\partial_m \mathcal{A}} \left[(\boldsymbol{\varphi}^0_{,1} \cdot \boldsymbol{\tau}^0)\phi_{I,2} - (\boldsymbol{\varphi}^0_{,2} \cdot \boldsymbol{\tau}^0)\phi_{I,1} \right] (\delta\boldsymbol{\mathcal{U}}_I \cdot \boldsymbol{t}^0)\lambda_\tau \ \mathrm{d}s \end{aligned} \tag{11.6}$$

11.3 Local coordinate system on the boundary

For reasons described in the section 6.6, it is desirable, that the local coordinate system basis on the boundary (i.e. the three vectors $\boldsymbol{\nu}^0$, $\boldsymbol{\tau}^0$ and $\boldsymbol{t}^0$) be as close to the exact basis as possible. Therefore, the following approach was adopted: If the surface in question is of simple form (e.g. a quadric), the surface normal is computed from the exact definition of the surface. Also, if the boundary curve is planar, the exact normal to the plane is used to make the local coordinate system as accurate as possible.

12 Weight function

The EFG method leads to a "parameterized" formulation of the discrete problem, where the parameters are the sizes of the domains of influence of the EFG points. These domains can be of any shape, but circles are the most common ones (i.e.

isotropic weight functions). The radius of the support circle of the I^{th} point is given by the definition of the weight function. The weight function needs to be

$$\begin{array}{ll} \text{(i)} & \text{non-negative, and} \\ \text{(ii)} & \text{it must hold that} \quad w_I(\boldsymbol{\xi}) = w(\|\boldsymbol{\xi} - \boldsymbol{\xi}_I\|, R_I) \ , \end{array} \tag{12.1}$$

where R_I is the radius of the support of the I^{th} node.

The EFG method has been in the meantime presented with a variety of weight functions. The weight function chosen for the Kirchhoff-Love shells is the quartic spline because of the continuity of the function and of its derivatives. The spline can be put down as a function of the normalized distance r

$$w(r) = \begin{cases} (1 - 6r^2 + 8r^3 - 3r^4) & \text{for } 1 > r \geq 0, \\ 0 & \text{for } r \geq 1. \end{cases} \tag{12.2}$$

with the normalized distance r being

$$r = \frac{\|\boldsymbol{\xi} - \boldsymbol{\xi}_I\|\cdot}{R_I} \ . \tag{12.3}$$

The support radius could be computed from the arrangement of the EFG points within the domain, for instance by requiring the domain of an EFG point to include a certain number of adjacent EFG points.

The radius of the support domain affects the solution. It is in this manner, that the term "parameterized discrete problem" is to be understood. The size of the support can be arbitrary, provided that it is large enough to yield a regular matrix $\boldsymbol{A}(\boldsymbol{x})$ of equation (4.4). It must include sufficiently large number of EFG points – at least 6 points for quadratic basis p_j – which must not be located in a special pattern (conic section for quadratic basis).

Further, the larger the support domain, the higher order the approximation achieved (by including larger number of EFG points). There is a limit to it, however. Consider a weight function radius approaching infinity. The moving least squares approximant then degenerates to standard quadratic least squares scheme. Increasing the domain of influence also makes the computation more costly.

Consequently, there emerges the question whether there is an "optimal" support radius, and how to compute it. The issue was investigated in a previous paper on thin plates by the EFG method by the authors [12]. The technique in this paper was based on quadratic basis, and it was found that the ratio $\kappa = R/h$ (R the support radius, h the "grid size", which was taken equal to the spacing of the EFG nodes) showed two points at which higher accuracy of the solution can be obtained – $\kappa \approx 3.4$ and $\kappa \approx 3.9$. As the irregular grids required in general larger support, the value of $\kappa \approx 3.9$ had been adopted in [12]. It should be noted, that both h and R are measured in the parametric space, therefore the results of Krysl and Belytschko [12] apply also for shells. However, the present paper uses not only quadratic basis, but also a quartic basis. Higher order polynomial basis requires larger support to achieve

optimal accuracy, however, and consequently another value of optimal radius was sought for the quartic basis. Again the search was based on numerical experiments, which gave a single optimal value $\kappa \approx 6.1$ (with overall higher accuracy, however).

13 Membrane locking

The membrane locking appears in shells (and beams) in which the membrane action is coupled with bending. The reason for membrane locking appearance is due to different approximation orders for the membrane and bending strains, so that the actual inextensional bending becomes polluted by parasitic membrane forces.

Membrane locking is usually significantly moderated by high-order approximation. Consequently, membrane locking should be significantly ameliorated in the present numerical model by increasing the support of the EFG nodes, as this increases the order of the approximation (to some extent – note the discussion in section 12. Also, increasing the order of the polynomial basis p_j of equation (4.1) should have similar effect and indeed, quartic basis practically removes membrane locking.

These theoretical consideration have been confirmed by numerical experiments (e.g. inextensional bending of cylindrical shell) as shown in the following section.

14 Numerical examples

14.1 Discretization

The discretizations of the examples considered in this section were constructed in the following manner: Geometric subdivisions of the domains in the form of quadrilaterals were used to define the shell surface by the above described moving least squares procedure. The EFG points were generated at the vertices of these quadrilaterals. The grids were regular or quasi-regular (for the hemispherical shell). It should be noted, that the present method associates three degrees of freedom (DOF) to a node, while most finite element models possess five or six DOF's per node. Therefore, the number of displacement DOF's is in the EFG models by 40 or 50% smaller than in finite element models with the same spacings between nodes.

The numerical quadrature was performed on the quadrilaterals by $N_G \times N_G$ Gaussian integration. The quadrature order was adopted as $N_G =?$ basing this decision on previous experiments in [12].

The essential boundary conditions have been enforced by the Lagrange multiplier method. The Lagrange multipliers were defined at the locations of the EFG points on the boundary. Linear interpolation was used on the boundary between the EFG points. One-point quadrature has been applied on the spans.

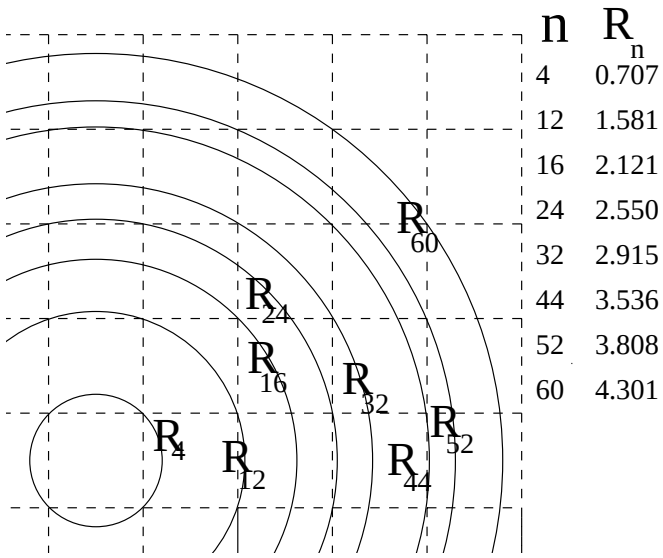


Figure 7: Values of parameter κ on a regular grid (square integration domains) for circles holding 4, 12, 16, 24, 32, 44, 52, and 60 EFG points.

14.2 Interpretation of the results

The support domains of all the EFG points were the same, so that we have $R_I = R$. Let us define a parameter κ given by

$$\kappa = \frac{R}{h} \, , \tag{14.1}$$

with R being the support radius (radius of the circle in which the shape function associated to an EFG node is non-zero), and h is the "mesh" size. The mesh size is for regular rectangular grids identical to the length of the longer side of the quadrature domain. In cases where the quadrature domains are of different shape, the mesh size h will be explicitly defined.

The results are dependent to some extent on the sizes of the domains of influence. Therefore, the results will be given with an indication of the support size that was used to compute them. As have shown previous experiments for plates, there are some "higher accuracy" support sizes. These issues have been discussed in section 12.

To help in the interpretation of the results, figure 7 presents an overview of the values of the parameter κ for regular grid composed of square integration domains with respect to the number of EFG points included in the circle of radius R (the EFG nodes are located at the vertices of the grid).

The test problems below are based on the MacNeal-Harder benchmarks as modified in the obstacle course of Belytschko *et al.* [4]; extensive details on these problems can be found in the latter reference.

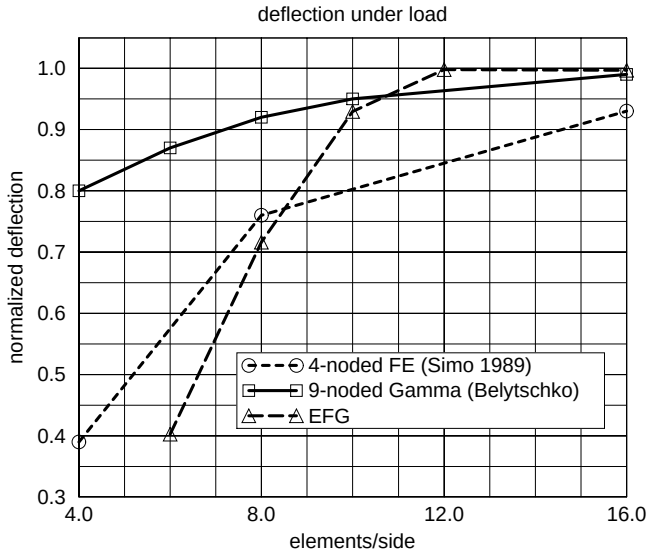


Figure 8: Pinched cylinder. Convergence of center deflection.

14.3 Pinched cylinder

The structure is a free cylinder loaded by a pair of pinching loads $P = 1$. The three symmetry planes provide the essential boundary conditions. The cylinder boundary curves are supported by a diaphragm which is flexible out of plane, but rigid in its plane. The cylinder length and radius are $L = 600$ and $R = 300$, respectively. The thickness is $t = 3$. The material properties are $E = 3 \times 10^6$ and $\nu = 0.3$. (All data in consistent physical units.) The analytical solution gives for the deflection under the load the value 1.82488×10^{-5}, which is used to normalize the numerical results in figure 8. The parameter κ was adopted in this computation as $\kappa = 6.1$, and the size h was equal to $(\pi R)/(2M)$, with M being the number of grid spacings along each side. The results are given for the quartic polynomial basis. It can be seen that the performance of the EFG technique compares well with high-performance finite elements, although worse results have been obtained for coarser grids (partly due to the fact that the grids did not contain sufficient number of EFG nodes to produce approximation of required order).

14.4 Scordelis-Lo barrel vault

The Scordelis-Lo barrel vault is a short cylindrical section loaded by gravity forces. The membrane and bending energies are almost equal (within 2% difference). Therefore, the membrane response is significant in this problem. The length of the cylinder is $L = 50$, radius $R = 25$, thickness is $t = 0.25$ and the span angle of the section is

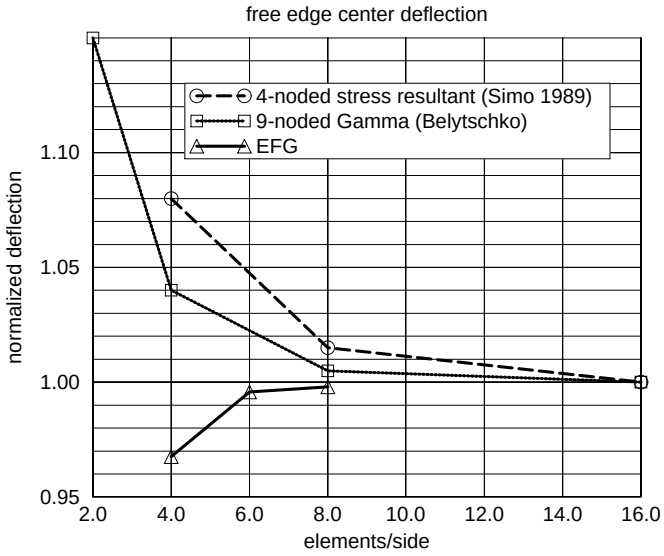


Figure 9: Scordelis-Lo barrel vault.

$\varphi = 80°$. Material properties are: $E = 4.32 \times 10^8$ and $\nu = 0$. The parameter κ was adopted in this computation as $\kappa = 6.1$, and the grid size h was equal to $(\pi R)/(2M)$, with M being the number of grid spacings along each side. There is a converged numerical solution of magnitude 0.3024 for the vertical deflection of the center of the free edge, which was used to normalize the results in figure 9. The polynomial basis adopted for this problem was quadratic as the shell is relatively flat (and the quartic polynomial basis is applied mainly to reduce membrane locking in bending-dominated cases). The present EFG method can be seen to give very satisfactory results in comparison to high-performance elements of Belytschko [4] and Simo [21].

14.5 Hemispherical shell

The hemispherical shell problem appears in the computational literature in two varieties: A shell with an opening at the top, and full hemisphere. The present results were obtained for the latter as in Belytschko *et al.* [4] (the problem thus becomes more severe as the quadrilaterals are distorted out-of-plane). The shell is pinched by two pairs of forces – in- and outward directed – of magnitude $P = 2$. The material constants are: $E = 6.825 \times 10^7$ and $\nu = 0.3$. The sphere radius is $R = 10$, and the thickness $t = 0.04$. The displacements in the direction of loads are the same, and the analytical value of 0.0924 was used to normalize the numerical results. The parameter κ was adopted in this computation as $\kappa = 6.1$, and the size h was equal to $(\pi R)/(2M)$, with M being the number of grid spacings along each side. Results are given for EFG method with (i) quadratic basis, (ii) quadratic basis and reduced

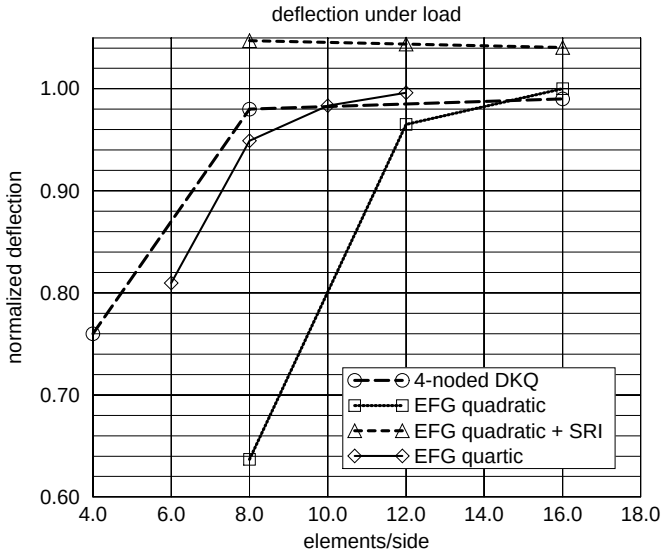


Figure 10: Hemispherical shell.

integration of the membrane stiffness (plot marked SRI), and (iii) quartic basis. It can be seen that the quartic basis produces faster convergence than the quadratic one, and also it can be concluded that the reduced integration is of dubious value here (also with respect to the use of the present technique in cracked shells, where the membrane response is very important, and should not be underintegrated). The relatively poor results for coarser grids are partly due to the fact that the grids did not contain sufficient number of EFG nodes to produce approximation of required order.

14.6 Inextensional bending of cylinder

This problem shows the behaviour of the present numerical model in relation to the membrane locking. The exact solution is based on inextensional bending of cylindrical shells, and is given by a series in the Timoshenko-Krieger monograph [24] (p. 432). The setup: Unsupported short cylinder is pinched by concentrated forces. Due to symmetry, only one eighth of the structure needs to be considered with appropriate boundary conditions on the planes of symmetry. Displacement under the force is obtained as $0.149(2Pa^3)/(\pi Dl)$, and horizontal displacement (increase in radius) is $0.137(2Pa^3)/(\pi Dl)$, with P being the force, a the radius of the cylinder, l its length, and $D = (Et^3)/(12(1-\nu^2))$ the bending stiffness.

Figure 11 shows the dependence of the solution accuracy on the support radius for a regular grid 9×9 EFG nodes. The size h was equal to $(\pi R)/(2M)$, with $M = 8$ being the number of grid spacings along each side. It can be seen, that the increase

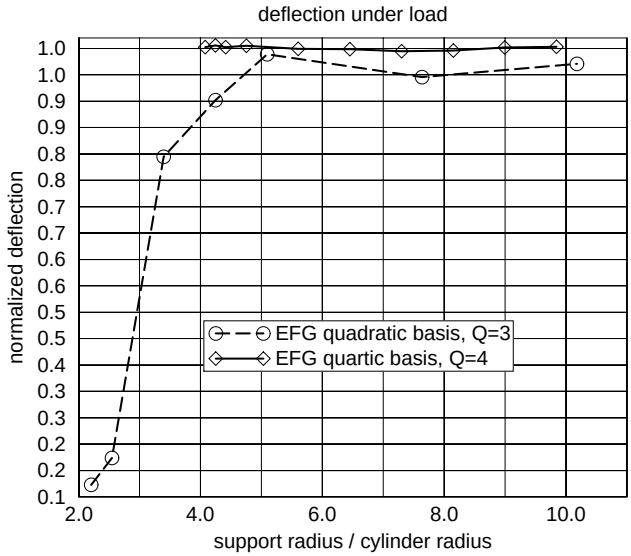


Figure 11: Inextensional bending of a cylinder.

in support radius alleviates membrane locking for the quadratic basis p_j, and use of quartic basis removes membrane locking altogether. The tags $Q = 3$ and $Q = 4$ in this figure give the number of quadrature points.

REMARK: The fact that the poor solutions really were due to membrane locking can be deduced from energy partitioning. The main part of the energy should be stored in this case in bending, the poor solutions possessed large share of membrane energy, however. (For the quadratic basis, membrane energy accounted for 76.81% for $\kappa = 2.2$ in comparison with 2.25% for $\kappa = 5.1$, and for the quartic basis membrane energy accounted for less than 2.0% for all ratios κ).

15 Conclusions

The Element-Free Galerkin (EFG) method has been applied to thin (Kirchhoff) shells. Isotropic material law and uniform shell thickness were assumed for simplicity, the results apply directly to any material law and any thickness variation, however.

The domain has been covered by a set of simple subdomains (background elements) for the purposes of surface shape approximation and also of numerical integration. Quadrilaterals were selected for the numerical implementation in this work; the geometric subdivision is immaterial, however, and any covering of the domain would do. The EFG nodes have been generated at the vertices of the geometric subdivision.

The shape of the surface has been approximated by the moving least squares technique from the vertices of the background mesh. An algorithm for the automatic parameterization of the background mesh has been proposed. Numerical integration was carried out on the background elements by Gaussian quadrature. A quadrature at 6×6 integration stations was adopted.

The polynomial basis used is a complete polynomial of second degree in the spatial coordinates. Therefore, consistency was achieved automatically. The resulting approximation is governed by the continuity of the weight function, which was adopted as a quartic spline. This function possesses requested C^1 continuity within the support, as well as on its boundary. In fact, due to the properties of the quartic spline weight function of (12.2), C^2 shape functions are constructed. The implications are that smooth internal resultants can be obtained without any re-interpolation or smoothing. Thus, while the finite element construction of C^1 numerical approximation is difficult and unsatisfactory so far, and while various devices to avoid the need for C^1 *ab initio* are employed (discrete Kirchhoff theory, hybrid stress, or even transition to C^0 theory), the current moving least squares method achieves C^1 approximation in a very straightforward manner.

The essential boundary conditions were enforced by Lagrange multipliers. One-point quadrature was applied along the spans between the EFG nodes on the supported boundaries. This is not the ideal method; however, more efficient and versatile techniques are under concurrent development.

The high accuracy and versatility of the present numerical approach have been demonstrated on a number of examples from the standard obstacle course for shells from [4]. The EFG method is flexible with respect to the construction of the shape functions. Therefore, it is possible to optimize the accuracy of the method by the choice of the weight function, by the selection of the support of the EFG nodes (given by the weight function definition). It was demonstrated that the method yields good results for quadratic polynomial basis. The membrane locking which appears in the numerical model was alleviated by enlarging the domains of influence of the EFG nodes for the quadratic basis, and it was removed completely by using quartic polynomial basis.

Acknowledgments

We gratefully acknowledge the support of the Office of Naval Research.

References

[1] T. Belytschko, L. Gu, and Y. Y. Lu. Fracture and crack growth by element-free Galerkin methods. *Modelling Simul. Mater. Sci. Eng.*, 2:519–534, 1994.

[2] T. Belytschko, Y. Y. Lu, and L. Gu. Element-free Galerkin methods. *International Journal of Numerical Methods in Engineering*, 37:229–256, 1994.

[3] T. Belytschko, D. Organ, and Y. Krongauz. A coupled finite element–element-free Galerkin method. *Computational Mechanics*, in press.

[4] T. Belytschko, H. Stolarski, W. K. Liu, N. Carpenter, and J. S-J. Ong. Stress projection for membrane and shear locking in shell finite elements. *Computer Methods in Applied Mechanics and Engineering*, 51:221–258, 1985.

[5] J. Bunch, L. Kaufman, and B. Parlett. Decomposition of a symmetric matrix. *Numerische Mathematik*, 27:95–109, 1976.

[6] C. R. Calladine. *Theory of shell structures*. Cambridge University Press, Cambridge; New York, 1983.

[7] W. S. Cleveland. *Visualizing data*. AT&T Bell Laboratories, Murray Hill, N.J., 1993.

[8] P. Hein. Diffuse element method applied to Kirchhoff plates. Technical report, Dept. Civil Engrg, Northwestern University, Evanston, Il., 1993.

[9] E. J. Kansa. Multiquadrics – a scattered data approximation scheme with applications to computational fluid dynamics: I. Surface approximations and partial derivative estimates. *Computers and Mathematics with Applications*, 19:127–145, 1990.

[10] E. J. Kansa. Multiquadrics – a scattered data approximation scheme with applications to computational fluid dynamics: II. Solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers and Mathematics with Applications*, 19:147–161, 1990.

[11] Y. Krongauz and T. Belytschko. Enforcement of essential boundary conditions in meshless approximations using finite elements. *Computer Methods in Applied Mechanics and Engineering, submitted.*, 1995.

[12] P. Krysl and T. Belytschko. Analysis of thin plates by the Element-Free Galerkin method. *Computational Mechanics*, submitted.

[13] P. Lancaster and K. Salkauskas. *Curve and surface fitting: an introduction*. Academic Press, London, Orlando, 1986.

[14] T. Liszka. An interpolation method for irregular net of nodes. *International Journal of Numerical Methods in Engineering*, 20:1599–1612, 1984.

[15] W. K. Liu, Sukky Jun, S. Li, J. Adee, and T. Belytschko. Reproducing kernel particle methods for structural dynamics. *International Journal of Numerical Methods in Engineering, accepted for publication.*, 1995.

[16] Y. Y. Lu, T. Belytschko, and L. Gu. A new implementation of the element free Galerkin method. *Computer Methods in Applied Mechanics and Engineering*, 113:397–414, 1994.

[17] J. J. Monaghan. An introduction to SPH. *Computer Physics Communications*, 48:89–96, 1982.

[18] B. Nayroles, G. Touzot, and P. Villon. Generalizing the finite element method: diffuse approximation and diffuse elements. *Computational Mechanics*, 10:307–318, 1992.

[19] S. Qian and J. Weiss. Wavelet and the numerical solution of partial differential equations. *Journal of Computational Physics*, 106:155–175, 1993.

[20] J. C. Simo and D. D. Fox. On a stress resultant geometrically exact shell model. part i: Formulation and optimal parametrization. *Computer Methods in Applied Mechanics and Engineering*, 72:267–304, 1989.

[21] J. C. Simo, D. D. Fox, and M. S. Rifai. On a stress resultant geometrically exact shell model. part ii: The linear theory. *Computer Methods in Applied Mechanics and Engineering*, 73:53–92, 1989.

[22] H. Stolarski, T. Belytschko, and S-H. Lee. A review of shell finite elements and corotational theories. *to be published*, 1995.

[23] W. G. Strang and G. J. Fix. *An analysis of the finite element method.* Prentice-Hall, Englewood Cliffs, N.J., 1973.

[24] S. Timoshenko and S. Woinowsky-Krieger. *Theory of plates and shells, 2d ed.* McGraw-Hill, New York, 1959.