Computer-Aided Design ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Curvilinear mesh generation using a variational framework☆

Michael Turner [a], Joaquim Peiró [a], David Moxey [b,*]

[a] Department of Aeronautics, South Kensington Campus, Imperial College London, London SW7 2AZ, UK
[b] College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, EX4 4QF, UK

ARTICLE INFO

Keywords:
High-order mesh generation
Variational mesh generation
Energy functional
Numerical optimisation

ABSTRACT

We aim to tackle the challenge of generating unstructured high-order meshes of complex three-dimensional bodies, which remains a significant bottleneck in the wider adoption of high-order methods. In particular we show that by adopting a variational approach to the generation process, many of the current popular high-order generation methods can be encompassed under a single unifying framework. This allows us to compare the effectiveness of these methods and to assess the quality of the meshes they produce in a systematic fashion. We present a detailed overview of the theory and formulation of the variational framework, and we highlight how such formulation can be effectively exploited to yield a highly-efficient parallel implementation. The effectiveness of this approach is examined by considering a number of two- and three-dimensional examples, where we show how the proposed approach can be used for both mesh quality optimisation and untangling of invalid high-order meshes.

© 2017 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

High-order methods are rapidly increasing in popularity due to their favourable numerical characteristics and ability to more effectively use modern computing hardware than traditional low-order methods. There has been much development of the underlying solvers, which give the ability to simulate fluid flows, acoustic phenomena and many other physical processes. However, these solvers ultimately rely on the partitioning of a domain into elements which, at high polynomial orders, must be: coarse in order to take advantage of the high-order nature of the method; curved to align with the underlying geometry; and valid, so that they do not self-intersect. The lack of development in this area has meant that this is a significant bottleneck in the more widespread use of these methods [1,2]. This is particularly applicable to industrial cases, where complex three-dimensional geometries representing (for example) cars and planes are clearly of significant interest. For these methods to become more popular outside of academia, this bottleneck clearly needs to be addressed.

Research in this area has mostly centred around *a posteriori* approaches, whereby a coarse linear mesh is deformed to accommodate the curvature at the boundary, and is the focus of this study. The challenge in this approach is to determine a method through which this curvature can be incorporated into the interior of the domain. Without this, the mesh is at best of a low quality, and at worst, will self-intersect, rendering it unsuitable for solver-based calculations. Existing work in *a posteriori* generation has broadly centred around two lines of investigation. The first of these focuses around the concept of solid body deformation, whereby the mesh is treated as a solid body which is deformed to incorporate curvature at the boundary. The work in this theme has focused around determining which model is 'best', either in terms of optimal quality or computational efficiency. Some models investigated include linear elasticity by Xie et al. [3] and Hartmann & Leicht [4], non-linear hyperelasticity by Persson & Peraire [5] and more recently by Poya et al. [6], thermo-elasticity by some of the authors of this work [7] and the Winslow equations by Fortunato & Persson [8]. The second theme follows a different route, whereby the mesh is equipped with an associated functional that denotes its energy. A non-linear optimisation problem is then solved in order to minimise this functional and yield a valid mesh. Again, most studies in this area have focused around this choice of functional, which include scaled Jacobian distortion metrics by Dey et al. [9], spring analogies for surface deformation by Sherwin & Peiró [10], unconstrained optimisation of the Jacobian by Toulorge et al. [11] and a number of articles by Roca and collaborators based on a shape distortion metric, e.g. [12–14].

However, what has so far remained unexplored in this area is the connections between these two themes. In the linear mesh generation community, for example in work by Garanzha [15] and Huang & Russell [16], it is known that through the calculus of variations, the elliptic partial differential equations defining these elasticity models can be recast into the minimisation of a energy functional, which takes as its arguments the mesh displacement and its derivatives. However, the use of this approach in high-order

☆ This paper has been recommended for acceptance by Chennai Guest Editor.

* Corresponding author.
E-mail address: d.moxey@exeter.ac.uk (D. Moxey).

https://doi.org/10.1016/j.cad.2017.10.004
0010-4485/© 2017 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

mesh generation has remained mostly unnoticed, asides from brief remarks in work by Sastry et al. [17].

The purpose of the present study is to examine the connections between these two existing mesh generation themes, by both recasting solid body models in a variational setting and examining other functionals already noted above in the literature. We will show that this approach has several benefits. It first allows us to examine each of the models in a common setting and investigate the relative benefits of each model in turn. Additionally, from a standpoint of robustness, the use of an energy functional that is convex or polyconvex, as investigated by Huang & Russell [16] and Garanzha [18], gives mathematical guarantees of a minimum that may be found using a numerical optimisation procedure. Finally, we note that early work in the 1970s by Felippa [19], who investigated direct energy minimisation methods for mesh generation, concluded that this method is promising but computational power was, at the time, a significant limiting factor in the success of this approach. In the following sections, we will show that modern computing hardware, combined with a suitable choice of numerical optimisation to exploit the denser structure that arises through a high-order discretisation, allows us to overcome this problem. The results we present here highlight that the variational setting allows us to construct a highly efficient and robust parallel framework for high-order mesh generation, permitting the generation of very complex three-dimensional meshes in the order of minutes.

Finally, we note that the groundwork for this study has been outlined in an earlier proceeding [20]. In this article we significantly expand the scope of the work by investigating several additional contributions. These are: the incorporation of optimisation procedures based on analytic gradients and Hessian regularisation; the implementation of an improved regularisation method used to untangle meshes and a detailed discussion of its properties; the extension of the method to permit the mesh nodes connected to the CAD geometry to slide along the curves and across the surfaces on the boundary; and, finally, the inclusion of a wider range of examples, including hybrid prismatic–tetrahedral boundary layer meshes and very high-order quadrilateral meshes.

The paper is structured as follows. Section 2 outlines the formulation of the problem in terms of a solid mechanics analogy. The four energy functionals that we will investigate in this work are introduced in Section 2.1, which overlap with a large number of studies based around high-order mesh generation, and we discuss a regularisation strategy to untangle invalid meshes in Section 2.2. Section 3 describes details of the practical implementation needed in this variational setting. This includes the discretisation and nonlinear optimisation in Sections 3.2 and 3.3, parallelisation strategies in Section 3.4 and allowing surface elements to slide across the CAD geometry in Section 3.5. Section 4 provides a brief analysis of the behaviour of the functional, guiding the choice of some numerical options. Section 5 then examines the application of this method to a number of two- and three-dimensional problems, describing the meshes obtained by each method, the number of iterations and computational time needed for convergence. We finalise the paper in Section 6 with a brief overview and outlook to future work and improvements.

2. Background and formulation

We begin with a brief mathematical overview of the setup of the variational formulation. The ultimate goal is to define an energy functional that will be optimised in order to produce a valid high-order mesh. We therefore first require a coarse mesh $\Omega_l = \bigcup_{e=1}^{N_{\text{el}}} \Omega_l^e$ of N_{el} straight-sided elements. The generation of this coarse grid is beyond the scope of this article but is discussed further in, e.g. Ref. [21]. We equip each element of Ω_l with a high-order polynomial finite element basis, based on standard Lagrange

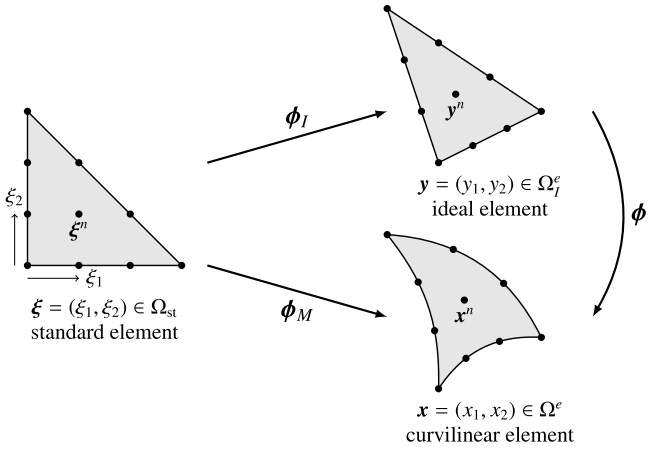


Fig. 1. Notation for mappings used throughout the paper: a triangular element is used for illustration purposes, but the notation is general and applicable to other element types. On the left we map a standard (reference) element Ω_{st} onto the straight-sided element Ω_l^e through the mapping $\boldsymbol{\phi}_l : \Omega_{\text{st}} \to \Omega_l^e$ and onto the curvilinear element $\boldsymbol{\phi}^e : \Omega_{\text{st}} \to \Omega^e$. The deformation mapping $\boldsymbol{\phi} : \Omega_l^e \to \Omega^e$ is then defined through the composition $\boldsymbol{\phi} = \boldsymbol{\phi}_M^e \circ \boldsymbol{\phi}_l^{-1}$.

interpolant basis functions. This gives an initial representation of the domain and serves as the initial configuration for the variational setup. In common with previous approaches [11,14], we define the mapping between a straight-sided mesh Ω_l and a curvilinear mesh Ω, which we subsequently denote by $\boldsymbol{\phi} : \Omega_l \to \Omega$. We refer to each element Ω_l^e as the 'ideal' element as it represents the best quality attainable without the introduction of curvature.

The mapping $\boldsymbol{\phi}$ is constructed by considering each element Ω_l^e separately. We refer to the diagram in Fig. 1, wherein we consider a triangular element and denote the coordinates inside each element as $\boldsymbol{\xi} \in \Omega_{\text{st}}$, $\boldsymbol{x} \in \Omega^e$ and $\boldsymbol{y} \in \Omega_l^e$. These mappings are constructed in an isoparametric fashion, so that the nodes $\boldsymbol{\xi}^n$ that define the Lagrange basis functions on the standard element map to $\boldsymbol{y}^n$ under $\boldsymbol{\phi}_l$ and $\boldsymbol{x}^n$ under $\boldsymbol{\phi}_M$. We note that other element types, such as quadrilaterals in two dimensions and tetrahedra, triangular prisms, pyramids and hexahedra in three dimensions, may use exactly the same definitions as above.

The energy functional is then defined as the integral

$$\mathcal{E}(\nabla\boldsymbol{\phi}) = \int_{\Omega_l} W(\nabla\boldsymbol{\phi})\,\mathrm{d}\boldsymbol{y}, \tag{1}$$

where W depends on the deformation gradient tensor

$$\nabla\boldsymbol{\phi}(\boldsymbol{y}) = \frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{y}}; \qquad [\nabla\boldsymbol{\phi}(\boldsymbol{y})]_{ij} = \frac{\partial\phi_i}{\partial y_j},$$

and its determinant $J = \det\nabla\boldsymbol{\phi}$, which we hereafter refer to as the Jacobian. In the following section we describe the different forms of the energy that we investigate in this article.

2.1. Forms of the energy functional

This section outlines a key contribution of this work, where we show that many of the existing curvilinear mesh generation methods can be unified in a variational setting through the definition of an energy functional. More importantly, a judicious choice of an energy functional that satisfies the convexity requirements of Ball's existence theory [22] guarantees the existence of a minimiser. We therefore seek to employ energy functionals that are *polyconvex*. A discussion of the properties of such functionals and how to verify them, together with examples of their use in mesh generation can be consulted in section 6.2 of the book by Huang and Russell [16].

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.

Further information of polyconvex energy functionals, their properties and applications to finite elasticity can be found, for instance, in the articles [23,24] and references therein. Similar ideas are also being explored in image registration [25].

2.1.1. Linear elasticity energy

A number of articles have examined the use of a linear elastic analogy in the context of high-order mesh generation [26,4,3]. This takes the form of an elliptic PDE

$$\nabla \cdot (\lambda \mathrm{tr}(\boldsymbol{E})\boldsymbol{I} + \mu \boldsymbol{E}) = -\boldsymbol{f}$$

where $\boldsymbol{E} = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top)$, $\boldsymbol{u} = \boldsymbol{x} - \boldsymbol{y}$ is the displacement from the straight-sided mesh configuration and λ and μ are the Lamé constants. Following common engineering practice, we work instead with Young's modulus, $E = \mu(3\lambda + 2\mu)/(\lambda + \mu)$, and Poisson's ratio, $\nu = \frac{1}{2}\lambda/(\lambda + \mu)$. Under these assumptions, a constant value of E is just a scaling factor of the body forces and the energy is a function of ν only. The Poisson's ratio is representative of the compressibility of the material with a value $\nu = \frac{1}{2}$ corresponding to an incompressible material. We further assume that there are no body forces so that $\boldsymbol{f} = \boldsymbol{0}$ and displacements are prescribed at the boundary to close the problem.

In the previous references, the standard approach is to adopt the usual Galerkin finite element discretisation, by defining trial and test functions that are polynomial expansions on each element and continuous between elements, applying integration by parts, and finally solving a linear system of equations involving a stiffness matrix to yield the displacements. However, we may alternatively view the above PDE as the Euler–Lagrange equation of the functional (1), where W is given by

$$W = \frac{1}{2}\lambda[\mathrm{tr}(\boldsymbol{E})]^2 + \mu\, \boldsymbol{E} : \boldsymbol{E},$$

where the double product or Frobenius product of two tensors is defined as $\boldsymbol{A} : \boldsymbol{B} = \mathrm{tr}(\boldsymbol{A}^\top \boldsymbol{B})$. The calculus of variations shows that the minimisation of the functional yields the same solution as obtained through the PDE. We note that while the above form of W does lead to the linear elasticity formulation for small deformations, but it does not satisfy the growth condition that $W \to \infty$ when $J \to 0^+$, which is required to prevent the inversion of the mesh [27]. By defining $\kappa > 0$ as the bulk modulus, a modified version of this energy that performs better for large compressive strains, i.e. when $J \to 0^+$, is

$$W = \frac{\kappa}{2}(\ln J)^2 + \mu\, \boldsymbol{E} : \boldsymbol{E}. \tag{2}$$

2.1.2. Isotropic hyperelasticity energy

We consider a nonlinear hyperelastic formulation that aligns with the work described in Refs. [5] and [6]. If the material is isotropic, so that the constitutive behaviour is identical in any direction, then the energy must be a function of the invariants of the right Cauchy–Green tensor, $\boldsymbol{C}$, only [28], where in this setting $\boldsymbol{C} = \nabla \boldsymbol{\phi}^\top \nabla \boldsymbol{\phi}$. This is written as $W(\boldsymbol{C}(\boldsymbol{y}), \boldsymbol{y}) = W(I_1^{\boldsymbol{C}}, I_2^{\boldsymbol{C}}, I_3^{\boldsymbol{C}}, \boldsymbol{y})$ where the invariants of $\boldsymbol{C}$ are

$$I_1^{\boldsymbol{C}} = \mathrm{tr}(\boldsymbol{C}) = \nabla \boldsymbol{\phi} : \nabla \boldsymbol{\phi}; \quad I_2^{\boldsymbol{C}} = \mathrm{tr}(\boldsymbol{C}\boldsymbol{C}) = \mathrm{tr}(\boldsymbol{C}^\top \boldsymbol{C}) = \boldsymbol{C} : \boldsymbol{C};$$
$$I_3^{\boldsymbol{C}} = \det(\boldsymbol{C}) = J^2.$$

A simple case of isotropic hyperelastic material is the compressible neo-Hookean material, as considered in Ref. [5], and its strain energy is given by

$$W = \frac{\mu}{2}(I_1^{\boldsymbol{C}} - 3) - \mu \ln J + \frac{\lambda}{2}(\ln J)^2$$

where λ and μ are the material constants, which we select as in the linear elasticity above.

2.1.3. Winslow equation energy

The Winslow equations are second-order non-linear elliptic partial differential equations which are obtained by enforcing the computational coordinates to be harmonic. These have long been used in the smoothing of linear meshes and have recently been used in the application of optimisation and untangling of high-order meshes [8]. They can be recast into a variational format by again viewing them as the Euler–Lagrange equation of the functional (1) with

$$W = \frac{\|\nabla \boldsymbol{\phi}\|_f^2}{J} \tag{3}$$

as shown, for example, in Ref. [29]. Here $\|\boldsymbol{F}\|_f = \sqrt{\mathrm{tr}(\boldsymbol{F}\boldsymbol{F}^\top)} = \sqrt{\boldsymbol{F} : \boldsymbol{F}}$ denotes the Frobenius norm induced by the inner product.

2.1.4. Energy as a measure of distortion

The final functional we consider here is a shape distortion measure that has been used in both linear [30] and curvilinear [12–14] mesh generation. We define (1) using

$$W = \frac{\|\nabla \boldsymbol{\phi}\|_f^2}{d|J|^{2/d}} \tag{4}$$

where d is the dimension of the mesh. An interesting point, which to the best of our knowledge has not been noted elsewhere in the literature, is the similarity between this distortion measure and the Winslow functional. Whilst the denominator of Eq. (4) ensures a different result for 3D meshes, we note that in the presence of a positive Jacobian in 2D, Eqs. (3) and (4) differ by only a factor of 1/2 and are therefore equivalent for the purpose of optimisation.

2.2. Mesh untangling via Jacobian regularisation

Each of the functionals used to describe the energy of deformation behave asymptotically around $J = 0$, so that $W \to \infty$ and $\mathcal{E} \to \infty$ as $J \to 0^+$. This is a desirable and expected property which prevents the mesh from inverting. The solid mechanics analogy would be that this behaviour prevents the interpenetration of matter. However, all of these functionals have various undesirable properties when $J < 0$, not least of which is that the elasticity analogies are undefined due to the presence of the term $\ln(J)$. Therefore, they are unsuitable for any problem where the initial mesh configuration is invalid, or equivalently, if during the optimisation of the functional, an invalid state is examined. To overcome this limitation, we replace J in the formulae above with a regularised version J_R previously employed for linear meshes [15,31] and high-order studies [13]. This takes the form

$$J_R = \frac{1}{2}\left(J + \sqrt{4\delta^2 + J^2}\right) \tag{5}$$

where δ is a small regularisation parameter. In the case of invalid elements, J_R becomes very small but positive, meaning that quantities such as $\ln(J_R)$ remain well defined. Additionally, the use of this regularisation alters the profiles of the energy functionals such that their behaviour shifts from being asymptotic to something similar to exponential and thus very large in the presence of invalid or marginally valid elements. This drives the optimisation process to move nodes in the mesh away from small or negative Jacobian regions since they possess very large energies, and correcting the mesh if it were to start to become invalid.

Selecting an appropriate δ requires careful examination, since it can dramatically change the profile of J_R for invalid or near-invalid elements. In studying various cases, we have found that maintaining a small δ even when elements are valid tend to give

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.

the best performance in untangling invalid meshes and optimising quality. We select

$$\delta = \begin{cases} \sqrt{10^{-8} + 0.04(J_{\min})^2}, & \text{if } J_{\min} < 0 \\ 10^{-4}, & \text{otherwise} \end{cases}$$

in which $J_{\min}$ refers to the minimum value of the Jacobian[1] in the mesh. This choice follows the strategy employed by Garanzha [15], but replaces the use of a machine epsilon with a small constant.

We note that the Jacobian regularisation is designed to assist in dealing with invalid and very nearly-invalid elements. For example, if $J = 10^{-4}$ then with our selected parameters, $J_R = 1.618 \times 10^{-4}$; with $J = 10^{-3}$, we have that $J_R = 1.009 \times 10^{-3}$; and at $J = 10^{-2}$, J_R is essentially identical for the purposes of optimisation. The resulting meshes obtained through this optimisation are therefore nearly identical in the validity regions, even for reasonably low-quality elements.

3. Implementation of the framework

After outlining the variational framework and the energy functionals to be investigated, we consider the practical numerical implementation in the optimisation of the functional (1). For large meshes comprising millions of moving nodes, we require both an efficient nonlinear optimisation method, alongside a robust calculation of the elemental contributions to the functional. We outline these details in this section, alongside a simple parallelisation strategy that can be used to mitigate the overall computational cost on many-core machines.

We note that the implementation described here is part of a new high-order meshing tool, *NekMesh* [21], which is contained in the open-source *Nektar++* spectral/*hp* element framework [32].

3.1. Evaluation of the functional on a single element

The functional $\mathcal{E}(\nabla\boldsymbol{\phi})$ in Eq. (1) is defined across the domain Ω_l. Given the elemental composition of the mesh, we may therefore naturally break this down into a summation of elemental contributions, so that

$$\mathcal{E}(\nabla\boldsymbol{\phi}) = \sum_{e=1}^{N_{\text{el}}} \int_{\Omega_l^e} W(\nabla\boldsymbol{\phi})\,\mathrm{d}\boldsymbol{y}.$$

Practically, we now require a discretisation of the mapping $\boldsymbol{\phi}$ on each element, and integration rules which allow us to numerically calculate the integral above. We therefore opt to employ the mapping $\boldsymbol{\phi}_l^{-1}$, as shown in Fig. 1, so that these requirements can be fulfilled by using the standard element Ω_{st}. This then allows us to define $\boldsymbol{\phi}$ as the composition $\boldsymbol{\phi}_M \circ \boldsymbol{\phi}_l^{-1}$. The summation above may therefore be written as

$$\mathcal{E}(\nabla\boldsymbol{\phi}) = \sum_{e=1}^{N_{\text{el}}} \int_{\Omega_{\text{st}}} W\left[\nabla\boldsymbol{\phi}_M(\boldsymbol{\xi})\nabla\boldsymbol{\phi}_l^{-1}(\boldsymbol{\phi}_l(\boldsymbol{\xi}))\right]\det(\nabla\boldsymbol{\phi}_l)\,\mathrm{d}\boldsymbol{\xi}. \tag{6}$$

We now need to construct $\nabla\boldsymbol{\phi}_l^{-1}(\boldsymbol{y})$ and $\nabla\boldsymbol{\phi}_M(\boldsymbol{\xi})$ on a given element, which we describe in the following sections.

3.1.1. Ideal mapping

The mapping $\boldsymbol{\phi}_l(\boldsymbol{y})$ may be written analytically as a combination of linear finite element shape functions, which makes its construction straightforward. For example, the mapping for a triangle with vertices $\boldsymbol{v}^1$, $\boldsymbol{v}^2$ and $\boldsymbol{v}^3$ is given by

$$\boldsymbol{\phi}_l(\boldsymbol{\xi}) = (\boldsymbol{v}^2 - \boldsymbol{v}^1)\xi_1 + (\boldsymbol{v}^3 - \boldsymbol{v}^1)\xi_2 + 1 \Rightarrow \nabla\boldsymbol{\phi}_l = \begin{bmatrix}\boldsymbol{v}^2 - \boldsymbol{v}^1 & \boldsymbol{v}^3 - \boldsymbol{v}^1\end{bmatrix}.$$

We note that this expression is independent of $\boldsymbol{\xi}$, which is also the case for tetrahedra in three dimensions. The inverse of the mapping can therefore be computed once for each element.

A similar approach can be adopted for other element types. For example, a quadrilateral with vertices $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^4$ in anti-clockwise ordering is described by the mapping

$$\boldsymbol{\phi}_l(\boldsymbol{\xi}) = \frac{1-\xi_1}{2}\frac{1-\xi_2}{2}\boldsymbol{v}^1 + \frac{1+\xi_1}{2}\frac{1-\xi_2}{2}\boldsymbol{v}^2 + \frac{1+\xi_1}{2}\frac{1+\xi_2}{2}\boldsymbol{v}^3 + \frac{1-\xi_1}{2}\frac{1+\xi_2}{2}\boldsymbol{v}^3.$$

with derivative

$$\nabla\boldsymbol{\phi}_l(\boldsymbol{y}) = \frac{1}{4}\begin{bmatrix}(\xi_2-1)\boldsymbol{v}^1 + (1-\xi_2)\boldsymbol{v}^2 + (1+\xi_2)\boldsymbol{v}^3 - (1+\xi_2)\boldsymbol{v}^3 \\ (\xi_1-1)\boldsymbol{v}^1 - (1+\xi_1)\boldsymbol{v}^2 + (1+\xi_1)\boldsymbol{v}^3 - (1+\xi_2)\boldsymbol{v}^3\end{bmatrix}^\top.$$

In this case we have a clear dependence on $\boldsymbol{\xi}$ and thus the inverse mapping will depend on $\boldsymbol{y}$. In this case, the derivative must be evaluated at chosen points $\{\tilde{\boldsymbol{\xi}}^q\}_{q=0}^Q$ within the standard element that correspond to points $\{\tilde{\boldsymbol{y}}^q\}_{q=1}^Q$ within Ω_l, and then each 2×2 or 3×3 matrix (depending on the dimension of the element) that represents $\nabla\boldsymbol{\phi}_l(\boldsymbol{\xi}^n)$ is inverted to construct $\nabla\boldsymbol{\phi}_l^{-1}(\tilde{\boldsymbol{y}}^q)$. The choice of these points aligns with the quadrature points we select to evaluate the integral inside Eq. (6), which we discuss in Section 3.2.3.

3.1.2. Curvilinear mapping

For $\boldsymbol{\phi}_M$ we select the usual isoparametric mapping that is common to nodal spectral element discretisations, which maps N nodes $\boldsymbol{\xi}^n$ from the standard element to nodes $\boldsymbol{x}^n$ inside the curvilinear element. We note that N depends on both the element type and polynomial order; for example a triangle at polynomial order P has $N = \frac{1}{2}(P+1)(P+2)$. Combined with the Lagrange polynomial interpolants $\boldsymbol{\ell}_n$ this yields the expansion

$$\boldsymbol{\phi}_M(\boldsymbol{\xi}) = \sum_{n=1}^{N} \boldsymbol{x}^n \boldsymbol{\ell}_n(\boldsymbol{\xi}).$$

We now require two sets of points inside Ω_{st}. The first is the set $\{\boldsymbol{\xi}^n\}_{n=1}^N$ which defines the Lagrange interpolants, so that $\boldsymbol{\ell}_m(\boldsymbol{\xi}^n) = \delta_{nm}$, yielding an isoparametric mapping such that $\boldsymbol{\phi}_M(\boldsymbol{\xi}^i) = \boldsymbol{x}^i$. For triangles and tetrahedra, we select the α-optimised points that are discussed in the book [33]. Hexahedra and quadrilaterals use tensor products of one-dimensional Gauss–Lobatto quadratures. Finally, prisms use a tensor product of the triangular α-optimised points and the Gauss–Lobatto points. Under these choices, the nodal points between any adjacent elements are guaranteed to align. We do not consider pyramidal elements in this work.

The second set of points is the set $\{\tilde{\boldsymbol{\xi}}^q\}_{q=1}^Q$, on which we will evaluate the integrals in Eq. (6) and $\nabla\boldsymbol{\phi}_l^{-1}$. It is important to highlight that in general, $Q \neq N$ and that $\boldsymbol{\xi}^n$ and $\tilde{\boldsymbol{\xi}}^q$ are not collocated. However, the selection of these points is important since they will impact both the accuracy of the evaluation of relevant quantities and the computational efficiency of the scheme.

Following the usual approaches that are outlined in greater detail in Refs. [33] or [34] for example, we may define an $N \times N$ discrete derivative operator $\mathcal{D}_j$ for each coordinate direction j, so that for $\boldsymbol{\Xi} = [\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^N]^\top$,

$$\mathcal{D}_j(\boldsymbol{\phi}_M)_i(\boldsymbol{\Xi}) = \frac{\partial(\boldsymbol{\phi}_M)_i}{\partial\xi_j}(\boldsymbol{\Xi})$$

where $\boldsymbol{\phi}_M = [(\phi_M)_1, (\phi_M)_2, (\phi_M)_3]$ in three dimensions. The derivatives at the integration points $\tilde{\boldsymbol{\Xi}} = [\tilde{\boldsymbol{\xi}}^1, \ldots, \tilde{\boldsymbol{\xi}}^Q]$, are evaluated by combining the derivative operator with an $Q \times N$ operator $\mathcal{I}$

[1] Evaluated at the integration points, see Section 3.2.3.

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.

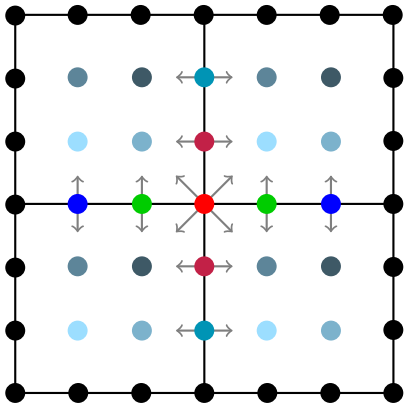


Fig. 2. Colouring scheme of 10 colours used to partition the mesh into sub-regions for parallel computing. Black vertices denote fixed (boundary) vertices that are not optimised and arrows show the elements that each vertex is connected to. Vertices that share the same colour can be optimised in parallel. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

which performs a polynomial interpolation from functions defined on $\boldsymbol{\xi}^p$ to $\tilde{\boldsymbol{\xi}}^q$ so that

$$\frac{\partial(\boldsymbol{\phi}_M)_i}{\partial \xi_j}(\tilde{\boldsymbol{\Xi}}) = \mathcal{I}\mathcal{D}_j(\boldsymbol{\phi}_M)_i(\boldsymbol{\Xi}).$$

Computational savings can be made by pre-computing the product of $\mathcal{I}$ and $\mathcal{D}_j$ for use in the following optimisation procedure.

3.2. Local optimisation procedure

As the energy functional is now defined and a method for its evaluation on a single element prescribed, we turn to the non-linear optimisation process that is required to minimise the deformation energy based on the selection of the integrand W in Eq. (1). In general, we wish to solve the minimisation problem

$$\text{find } \min_{\boldsymbol{\phi}} \mathcal{E}(\boldsymbol{\phi}) = \min_{\boldsymbol{\phi}} \int_{\Omega_l} W(\nabla\boldsymbol{\phi})\,\mathrm{d}\boldsymbol{y}. \tag{7}$$

We note that, implicitly, $\boldsymbol{\phi}_M$ and therefore $\boldsymbol{\phi}$ depend on the set of nodes that defines the curvilinear mesh, which are denoted by $\boldsymbol{N} = [\boldsymbol{n}^1, \ldots, \boldsymbol{n}^{N_{\text{nodes}}}]$ and each $\boldsymbol{n}^i$ represents the coordinates of a node in the mesh. The optimisation of this problem therefore begins with an initial selection of the nodes, $\boldsymbol{N}^0$, in which we take the linear mesh and impose the curvature of the boundary, regardless of whether this causes self-intersections.

We then require a numerical optimisation strategy to iteratively update $\boldsymbol{N}$ towards lower energy configurations, i.e. $\boldsymbol{N}^k \to \boldsymbol{N}^{k+1}$ such that $\mathcal{E}^k \to \mathcal{E}^{k+1}$ where $\mathcal{E}^{k+1} < \mathcal{E}^k$. To achieve this, we choose to adopt a gradient- and Hessian-based method as this can provide superior convergence properties. Following Ref. [14] and later works, we also adopt an optimisation method which is based on a relaxation strategy. In a 'standard' optimisation we would compute derivatives in a global fashion to highlight the effects of moving a node on all other nodes of the mesh. However, this requires the solution of a large matrix system. In the relaxation approach, we instead solve a cheaper local optimisation problem for each node $\boldsymbol{n}^i \in \boldsymbol{N}$ which takes the form

$$\text{find } \min_{\boldsymbol{\phi}} \mathcal{E}_i(\nabla\boldsymbol{\phi}) = \sum_{e\subset i} \int_{\Omega_l^e} W(\nabla\boldsymbol{\phi})\,\mathrm{d}\boldsymbol{y}. \tag{8}$$

Here, $e \subset i$ denotes the subset of elements influenced by a change in the position of node i. For example, Fig. 2 shows that if we adjust the position of nodes lying on the interior quadrilateral edges, we only need to consider the evaluation of a functional that is connected to either two or four elements as denoted by the arrows, and any interior nodes only need to be considered on a single element. The minimisation of (7) can then be considered as that of a non-linearly related set of energies, each of which belongs to a node and is evaluated using a subset of elements in the mesh. We note that these problems are far more computationally tractable: they are highly compact and memory efficient, amenable to parallelisation and have simpler expressions for gradient and Hessian terms. However they come at the cost of increased iteration counts over the global approach and can potentially become more prone to becoming stuck in local minima of the functional.

3.2.1. Numerical optimisation techniques

The optimisation of each $\mathcal{E}_i$ sub-problem, i.e. to find the minimum of $\mathcal{E}_i$, is described in algorithm 1 , which is split into two procedures. The LocalOptimisation procedure performs a standard gradient descent, where the search direction is improved through use of a Newton-based gradient and Hessian of $\mathcal{E}_i(\nabla\boldsymbol{\phi})$. This is coupled with a reverse line search, using one of two Wolfe conditions, with standard choices for the various parameters used to perform the line search [35]. We note that the second Wolfe condition, in which the gradient is re-evaluated at each iteration of the line search, is omitted, as we have found that the significant additional cost of gradient calculation does not yield any great improvement in robustness.

We now briefly discuss the choice of optimisation procedure. The convexity property of the functionals used in this article means that the Hessian of second derivatives is a symmetric positive definite (SPD) matrix, and therefore a truncated Newton type optimiser is the clear choice of method, as the algorithm will be able to exploit properties such as a superior rate of convergence. However, we do note that the SPD property is dependent on two conditions. Firstly, if the functional or its derivatives are not calculated to a sufficient degree of accuracy, we may encounter either very small or negative eigenvalues in the resulting matrix, which may significant impact on the choice of search direction. We investigate this further in Section 4. Barring this however, as noted by Garanzha [18], the use of regularisation compromises the convexity property of the functional. Indeed we see that as J becomes small and J_R begins to deviate significantly from J, the eigenvalues of the Hessian become very small or negative, and the matrix is quite commonly indefinite. In these cases, an indefinite Hessian will still frequently yield a new node position with lower energy. It also results in more 'reset' nodes in which no minima are found, as well as generally slower convergence speeds due to more functional evaluations arising from smaller values of the step-length parameter, α, in the line search.

We therefore opt to employ a commonly used Hessian regularisation technique, whereby the Hessian is altered if it contains small positive or any negative eigenvalues. Note that this is different to the Jacobian regularisation discussed in Section 2.2, in which the functional is altered mathematically: this technique is purely a numerical improvement. The Hessian regularisation takes the form

$$\boldsymbol{H} = \begin{cases} \boldsymbol{H} + (\beta - \lambda_{\min})\boldsymbol{I}, & \lambda_{\min} < \beta, \\ \boldsymbol{H}, & \text{otherwise}, \end{cases}$$

where we select $\beta = 10^{-6}$ and $\lambda_{\min}$ is the minimum eigenvalue of $\boldsymbol{H}$. The use of this regularisation forces the Hessian to become SPD and we have found greatly increases convergence speed and robustness. We observe that when untangling a mesh, the Hessian regularisation will be used almost exclusively within invalid elements in the early iterations. Once the mesh is sufficiently valid, i.e. when all Jacobians are positive and far from zero, where the Jacobian regularisation has no effect, the Hessian regularisation is no

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.

Algorithm 1 Solve the local optimisation problem for node $\boldsymbol{n}^i$ using functional defined by W. Returns new node location.

procedure LocalOptimisation(W, $\boldsymbol{n}^i$)
  $\boldsymbol{A} \leftarrow$ set of elements connected to $\boldsymbol{n}^i$
  $\mathcal{E}, \boldsymbol{G}(\mathcal{E}), \boldsymbol{H}(\mathcal{E}) \leftarrow$ EvaluateFunctional(W, $\boldsymbol{n}^i$, $\boldsymbol{A}$) ▷ evaluate functional, derivatives and Hessian
  $G_{\text{tol}} \leftarrow 10^{-10}$ ▷ tolerance for zero gradient
  if $\|\boldsymbol{G}(\mathcal{E})\| < G_{\text{tol}}$ **then**
    return ▷ nodes with zero gradient already optimal
  end if
  $\boldsymbol{m} \leftarrow \boldsymbol{n}^i$ ▷ temporary storage
  $\lambda \leftarrow$ minimum eigenvalue of $\boldsymbol{H}$
  if $\lambda < 10^{-6}$ **then**
    $\boldsymbol{H}(\mathcal{E}) \leftarrow \boldsymbol{H}(\mathcal{E}) + (10^{-6} - \lambda)\boldsymbol{I}$ ▷ Hessian regularisation
  end if
  $\boldsymbol{s}_k \leftarrow -\boldsymbol{H}(\mathcal{E})^{-1}\boldsymbol{G}(\mathcal{E})$ ▷ search direction for descent
  $\alpha, \alpha_{\text{tol}}, c_1 \leftarrow 1, 10^{-10}, 10^{-3}$ ▷ tolerances for line search
  $p \leftarrow \boldsymbol{s}_k^\top \boldsymbol{G}(\mathcal{E})$
  while $\alpha > \alpha_{\text{tol}}$ **do** ▷ reverse line search
    $\boldsymbol{n}^i \leftarrow \boldsymbol{m} + \alpha \boldsymbol{s}_k$ ▷ move node in the descent direction
    $\mathcal{F} \leftarrow$ EvaluateFunctional(W, $\boldsymbol{m}$, $\boldsymbol{A}$) ▷ evaluation functional only, note this is with modified node position
    if $\mathcal{F} \leq \mathcal{E} + c_1 \alpha p$ **then** ▷ using Wolfe condition
      return $\boldsymbol{n}^i$ ▷ new minimum found
    end if
    $\alpha \leftarrow \frac{1}{2}\alpha$
  end while
  return $\boldsymbol{m}$ ▷ unable to optimise, reset node
end procedure
procedure EvaluateFunctional(W, $\boldsymbol{n}$, $\boldsymbol{A}$)
  $\mathcal{E}, \boldsymbol{G}(\mathcal{E}), \boldsymbol{H}(\mathcal{E}) \leftarrow 0, \boldsymbol{0}, \boldsymbol{0}$ ▷ $\mathcal{E}$ is functional, $\boldsymbol{G}(\mathcal{E})[\cdot]$ is gradient vector, $\boldsymbol{H}(\mathcal{E})[\cdot,\cdot]$ is Hessian matrix
  for each type of element in $\boldsymbol{A}$ **do** ▷ e.g. triangle, quad
    $\boldsymbol{x} \leftarrow$ coordinates of all elements of this type ▷ $\boldsymbol{x}$ is a matrix of coordinates for these elements
    for each coordinate direction d **do**
      $\nabla \boldsymbol{x}_d \leftarrow \mathcal{D}_d^{\text{type}} \boldsymbol{x}_d$ ▷ compute deformation tensor in direction d using `dgemm`
    end for
    for each element Ω **do**
      for each evaluation point and weight $\tilde{\boldsymbol{\xi}}^i$, w_i inside Ω **do**
        $w \leftarrow w_i \det \boldsymbol{\phi}_l(\tilde{\boldsymbol{\xi}}^i)$ ▷ ideal mapping determinant weighted by quadrature weight
        $\mathcal{E} \leftarrow \mathcal{E} + W(\nabla \boldsymbol{x}) \cdot w$ ▷ evaluate functional using regularised Jacobian if required
        if not calculating gradient and Hessian **then**
          continue
        end if
        for each coordinate direction j **do**
          $\boldsymbol{G}(\mathcal{E})[j] \leftarrow \boldsymbol{G}(\mathcal{E})[j] + \partial_{n_j} W(\nabla \boldsymbol{x}) \cdot w$ ▷ analytic gradient evaluation
          for each coordinate direction k **do**
            $\boldsymbol{H}(\mathcal{E})[j,k] \leftarrow \boldsymbol{H}(\mathcal{E})[j,k] + \partial_{n_j n_k} W(\nabla \boldsymbol{x}) \cdot w$ ▷ analytic Hessian evaluation
          end for
        end for
      end for
    end for
  end for
end procedure

longer used by the optimisation algorithm because the convexity property has been restored.

We also note that, since the Hessian matrix is small – of size either 2×2 or 3×3 depending on the dimension of the problem – analytic expressions exist to calculate the minimum eigenvalue without the need for common linear algebra packages such as LAPACK. This is therefore a relatively cheap route to improving the robustness of the numerical optimisation.

3.2.2. Functional, gradient and Hessian evaluation

The LocalOptimisation procedure uses a separate EvaluateFunctional routine to evaluate the functional, as well as its gradient and Hessian, during the line search. A very important point to note is that these derivatives are evaluated *not* with respect to the usual Cartesian coordinate directions, but instead with respect to the *position of the node* $\boldsymbol{n}^i$. In our previous work [20], we evaluated these terms using a finite-difference approximation, where a stencil was used to approximate the gradient terms. Whilst this is very flexible, as it only requires the evaluation of W at each point of the stencil, it introduces additional parameters in terms of the physical size of the stencil to use. It is also computationally expensive, particularly in three dimensions where 13 stencil points are required to calculate a first-order approximation.

In this work, we have derived a series of analytic expressions for the gradient and Hessian in order to reduce the computational cost and improve robustness. A detailed derivation of these terms is long and detracts from the present discussion. For completeness however we have included a detailed derivation of these terms in

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.

B. These analytic expressions have been validated by comparing against the values obtained via the finite-difference approximation.

A final point to note regarding the evaluation of the gradient terms $\boldsymbol{\phi}_M(\boldsymbol{\xi})$ is that this operation may be greatly increased in speed using useful insights from previous work [36]. If we amalgamate the coordinates of all of these elements into one matrix, then $\boldsymbol{\phi}_M(\boldsymbol{\xi})$ can be calculated using either two or three matrix–matrix multiplications, depending on the dimension of the problem, where the left-hand matrix is the combined derivative and interpolation operator $\mathcal{D}_i$. We may therefore utilise an optimised BLAS call such as `dgemm`, which will give significantly higher performance but only at larger polynomial orders. We have found that in the presence of lower order elements, where matrix sizes are far smaller, BLAS can be far less efficient than hand-written loops. In this case we use the library `libxsmm` [37], which performs just-in-time compilation for small matrix–matrix multiplications to execute optimised hand-written kernels. We have found that this yields a significant performance improvement of up to around 25% for these problems.

3.2.3. Quadrature rules

One of the key aspects of the evaluation of elemental contributions is the quadrature rule used to approximate the integral of the functional across each element, which in earlier sections is denoted by the set $\{\tilde{\boldsymbol{\xi}}\}_{q=0}^{Q}$. Our initial efforts used high-order nodes conforming to an α-optimised set of points on triangles and tetrahedra [33]. Whilst this is a commonly used distribution and has been used in the optimisation of the shape distortion functional in Ref. [13], our experience has been that this distribution is prone to introducing instability as part of the nonlinear optimisation.

Closer observation shows that, at most polynomial orders, these distributions have negative quadrature weights at the vertices of the element. The presence of negative weights can induce instability in instances when a mesh node is moved to a position that results in a very high elemental deformation. This yields an abnormally large value of the functional at the element vertex which, when multiplied by a negative weight, results in a large negative value. In the summation over all quadrature points, this can then result in a very large negative value of the functional, which causes the optimiser to locate a new minimum where one does not exist.

A potential solution is the use of a larger number of quadrature points, i.e. a significant increase in Q, to evaluate the gradient of the deformation tensor and functional, as is performed in e.g. [13]. Although the quadrature weights are still negative at the vertices of the triangle, there is a greater clustering of quadrature points in these areas. This means that the large gradient of the deformation can be accurately resolved, thereby preserving positivity of the integral. Our testing showed that whilst this yielded additional stability it did come at a greatly increased computational cost.

To overcome both the cost and stability issues we propose using an alternative set of quadrature points for triangles, tetrahedra and prismatic elements. We use quadrature rules proposed by Witherden and Vincent [38], which are symmetric, interior to the standard element, but crucially have positive quadrature weights for all elements at all orders. These point distribution sets also have lower numbers of points which can achieve the same level of accuracy in integration. Therefore we no longer need to over-integrate to such a large extent, resulting in a lower computational cost whilst preserving the robustness of the method. This choice however is far from unique: there are many such quadrature rules in the literature which may also yield similar properties [39]. Finally, we note that for quadrilaterals and hexahedra, we opt to use Gauss–Lobatto points which are readily computed and have positive weights. Prismatic elements inherit both properties from the tensor product of triangular and Gauss–Lobatto points. We discuss the level of over-integration required in Section 4.

3.3. Global optimisation

Finally, we discuss the last part of the optimisation procedure: iterating over each of the local optimisation problem to converge to a global minimum, which is described in algorithm 2 . The first stage is straightforward: we compile a list $\boldsymbol{N}$ of the nodes to be optimised. Whilst nodes on the boundary may remain fixed (and thus be omitted from this list), they may also be allowed to move across the surface of the mesh, which we describe in the following section.

In terms of the optimisation process, one step of the GlobalOptimise procedure iterates over all local optimisation problems, which results in a decrease of the global functional value. These steps then continue until a convergence criterion has been met. We set a strong convergence criterion such that $\|\boldsymbol{N}^{k+1} - \boldsymbol{N}^{k}\|_{\infty}/L < \varepsilon$, so that the mesh is optimised once no node has moved more than a small fraction, ε, of a characteristic length, L, of the problem. We use $\varepsilon = 10^{-6}$ as a guideline for convergence. However other convergence criteria, such as a lower bound on the minimum Jacobian in the mesh, may be set if this is a more important outcome.

3.4. Parallelisation

A significant issue in the use of mesh optimisation and deformation strategies is that of efficient runtime. Simply put, the practical usefulness of the method relies in the end-user being able to run the optimisation procedure 'quickly', ideally in no more than a few minutes. To address the issue of high associated cost of the global optimisation, we briefly outline a parallelisation strategy to decrease execution time.

One of the main motivations and advantages of using a nonlinear optimisation based on a relaxation strategy is that it can effectively align to the requirements of modern computing hardware in order to decrease the optimisation runtime. To illustrate this, consider two nodes $\boldsymbol{n}^i$ and $\boldsymbol{n}^j$, which are connected to groups of elements $e \subset i$ and $f \subset j$. Then we observe that, by definition of the corresponding local functionals $\mathcal{E}_i$ and $\mathcal{E}_j$ which have support only on these element groups, the LocalOptimisation procedure for $\boldsymbol{n}^i$ and $\boldsymbol{n}^j$ can be executed concurrently on $\mathcal{E}_i$ and $\mathcal{E}_j$ if the elemental sets $\bigcup_{e\subset i}\Omega^e$ and $\bigcup_{f\subset j}\Omega^f$ do not intersect.

From a practical perspective, we need to solve two problems: determining sets of local optimisations that can be run in parallel, and choosing a parallelisation framework. To solve the first problem, we adopt a straightforward node colouring scheme, similar to that used in graph partitioning. In this setting, we assign colours to nodes such that if two nodes have the same colour, then their associated local optimisation problem can be run in parallel. An example of the node colouring scheme applied to a small sample mesh is shown in Fig. 2. The precise algorithm for node colouring is outlined in the ColourNodes routine, which implements a simple node colouring strategy to sort all nodes in the mesh that are not fixed into subsets which are disjoint from any other subsets, allowing each local optimisation to be run independently.

Given the node colourings, we tackle the solution of the second problem, i.e. the choice of a parallelisation framework, through the use of a `pthread` model. We note that this therefore precludes the use of distributed parallelism, as is commonly enabled through the use of MPI. However, in the use case we envision, most users will be constrained to a single workstation, and therefore threading provides an approach to exploit the shared memory of this setting without additional complexity of distributed parallelisation. In addition, the strategy is well suited for implementation in GPU architectures.

For each node colour, we assemble a list of local optimisation problems that can be viewed as a 'task'. We then start a number of threads, corresponding to the core count of the CPU being run,

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.

Algorithm 2 Solve the global optimisation problem for all nodes $\boldsymbol{N}$ with the functional defined by W.

```
procedure ColourNodes(N)
    while N is not empty do
        M ← any uncoloured nodes in N
        Create a new colour c
        while M is not empty do
            Select a node n ∈ M, colour n with c
            Let A ← be the set of elements connected to node n
            Remove n and all nodes in A from M
        end while
    end while
    return colours for each node
end procedure
procedure GlobalOptimise(N, W)
    call ColourNodes(N)
    M ← [0, ..., 0]_{n=1}^{N}                ▷ will store updated configurations from local optimisation
    while ‖N − M‖_∞ > 10^{−6} do
        for each colour c do
            for each node n with colour c do in parallel
                m^i ← LocalOptimisation(W, n)   ▷ store new node position
            end for
        end for
    end while
end procedure
```

and use a thread manager to run each 'task' in the parallel **for** loop of the GlobalOptimise routine. This is particularly efficient, since the node colouring guarantees that there are no inter-task dependencies. Each of the compute threads can therefore run concurrently with virtually no locks or mutexes, maximising use of the hardware. The effectiveness of this parallelisation, and in particular the strong scaling of the method, will be demonstrated in the results section.

3.5. Surface mesh optimisation

In the discussion so far, we have considered the boundary mesh vertices and high-order nodes to be fixed, which closes the system and forms a well-posed problem. However, in complex geometric examples it is quite possible that the surface mesh can have a low quality or even invalid configuration. Frequently this arises because of poor or complex parametrisation in the CAD, which is difficult to overcome *a priori* in the surface mesh generation process. If this is the case, fixing the nodes connected to the surface during optimisation will constrain the mesh to such a degree that the resulting mesh may be sub-optimal or invalid. This is especially concerning in the case of an invalid surface mesh, as it will prevent the mesh from ever becoming valid. To overcome this, should the CAD description of the geometry be available, the optimisation framework is capable of 'sliding' the surface mesh nodes to an optimised location while keeping them bound to their respective CAD parents. This follows the same themes established in earlier work by Sherwin & Peiró [10] and Ruiz-Gironés et al. [40].

We allow the surface mesh to slide by considering mesh nodes which exist on a CAD vertex to be fixed, in order to close the system. Nodes which belong to CAD curves are able to move in their one-dimensional parametric space, denoted by $\boldsymbol{x} = \boldsymbol{r}(t)$. Similarly, nodes on CAD surfaces are able to move in their two-dimensional parametric spaces $\boldsymbol{x} = \boldsymbol{r}(\boldsymbol{u}) = \boldsymbol{r}(u, v)$. To take this parametrisation into account in the optimisation process, the LocalOptimisation procedure must be adapted for nodes connected to CAD curve and surface nodes. Specifically, instead of moving the three coordinates of the node position $\boldsymbol{n}^i$, our variables become the curve and surface parameters t and (u, v) respectively. Through an application of the chain rule, the gradient and Hessian which are required during the optimisation routine are given by

$$\frac{\partial \mathcal{E}(\nabla\boldsymbol{\phi})}{\partial t} = \frac{\partial \mathcal{E}(\nabla\boldsymbol{\phi})}{\partial \boldsymbol{n}^i} \cdot \frac{\partial \boldsymbol{n}^i}{\partial t},$$
$$\frac{\partial^2 \mathcal{E}(\nabla\boldsymbol{\phi})}{\partial t^2} = \frac{\partial \mathcal{E}(\nabla\boldsymbol{\phi})}{\partial \boldsymbol{n}^i} \cdot \frac{\partial^2 \boldsymbol{n}^i}{\partial t^2} + \left(\frac{\partial \boldsymbol{n}^i}{\partial t}\right)^{\top} \cdot \frac{\partial^2 \mathcal{E}(\nabla\boldsymbol{\phi})}{\partial \boldsymbol{n}^{i2}} \cdot \frac{\partial \boldsymbol{n}^i}{\partial t},$$

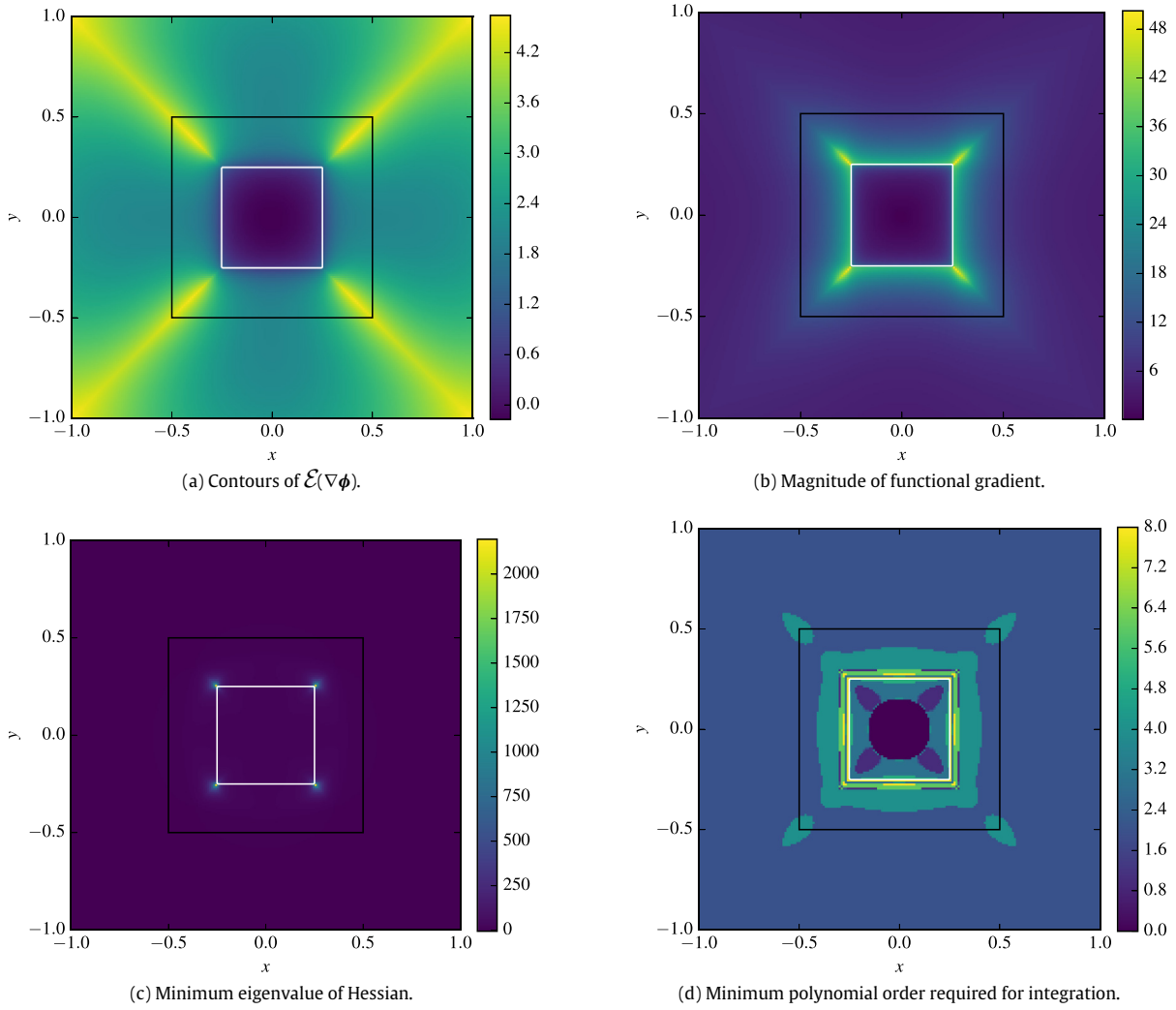
for curves, and

$$\frac{\partial \mathcal{E}(\nabla\boldsymbol{\phi})}{\partial \boldsymbol{u}} = \frac{\partial \mathcal{E}(\nabla\boldsymbol{\phi})}{\partial \boldsymbol{n}^i} \cdot \frac{\partial \boldsymbol{n}^i}{\partial \boldsymbol{u}},$$
$$\frac{\partial^2 \mathcal{E}(\nabla\boldsymbol{\phi})}{\partial \boldsymbol{u}^2} = \left(\frac{\partial \boldsymbol{n}^i}{\partial \boldsymbol{u}}\right)^{\top} \cdot \frac{\partial^2 \mathcal{E}(\nabla\boldsymbol{\phi})}{\partial \boldsymbol{n}^{i2}} \cdot \frac{\partial \boldsymbol{n}^i}{\partial \boldsymbol{u}} + \frac{\partial \mathcal{E}(\nabla\boldsymbol{\phi})}{\partial \boldsymbol{n}^i} \cdot \frac{\partial^2 \boldsymbol{n}^i}{\partial \boldsymbol{u}^2}.$$

We note that the derivatives in the parametric spaces that are required in the above expressions, such as $\partial_t \boldsymbol{n}^i$, are provided by the CAD engine: in our case, OpenCASCADE. We must also ensure that the parametric coordinates do not move to a position outside of their bounding box that defines the surface. With the exception of these two issues however, the core of the optimisation process remains the exactly the same as that used for the free-to-move nodes.

4. Analysis of the functional evaluation

In order to gain insight into the behaviour of the functional, both in terms of determining an appropriate integration order and of understanding how it will behave under optimisation, in Fig. 3 we examine the evaluation of the hyperelastic energy for a second order quadrilateral, which is centred at the origin and has sides of unit length. This is shown by the black box in these figures. We fix all the nodes of the quadrilateral, asides from the single interior degree of freedom located in the centre of the element. This node is moved to positions $(x, y) \in [-1, 1]^2$, where at each position we evaluate: the functional, $\mathcal{E}(\nabla\boldsymbol{\phi})$; the magnitude of the gradient of the functional, $\|\partial_{\boldsymbol{n}} \mathcal{E}(\nabla\boldsymbol{\phi})\|$ under the L^2 norm; the minimum eigenvalue of the Hessian matrix $\partial^2_{\boldsymbol{n}} \mathcal{E}(\nabla\boldsymbol{\phi})$; and an estimate of the over-integration required to evaluate the functional. These quantities are depicted as coloured contour plots in Fig. 3a, 3b, 3c and 3d, respectively. Finally, the white region in the centre of the element



(a) Contours of $\mathcal{E}(\nabla\boldsymbol{\phi})$.

(b) Magnitude of functional gradient.

(c) Minimum eigenvalue of Hessian.

(d) Minimum polynomial order required for integration.

Fig. 3. A quantitative analysis of the hyperelastic functional for a second-order quadrilateral.

highlights the level set $\min_{\boldsymbol{\xi}\in\Omega} J(\boldsymbol{\xi}) = 0$. This therefore highlights the region of validity of the element, so that if the interior node remains within this region, the element is valid, whilst outside it becomes invalid. We additionally recall that the choice of the quadrature points $\{\tilde{\boldsymbol{\xi}}^q\}_{q=1}^{Q}$ is different from the nodal points $\{\boldsymbol{\xi}\}_{n=1}^{N}$ used to define the basis functions of the element. In this second order quadrilateral, we use $N = 9$ nodal points and are left to select appropriate quadrature.

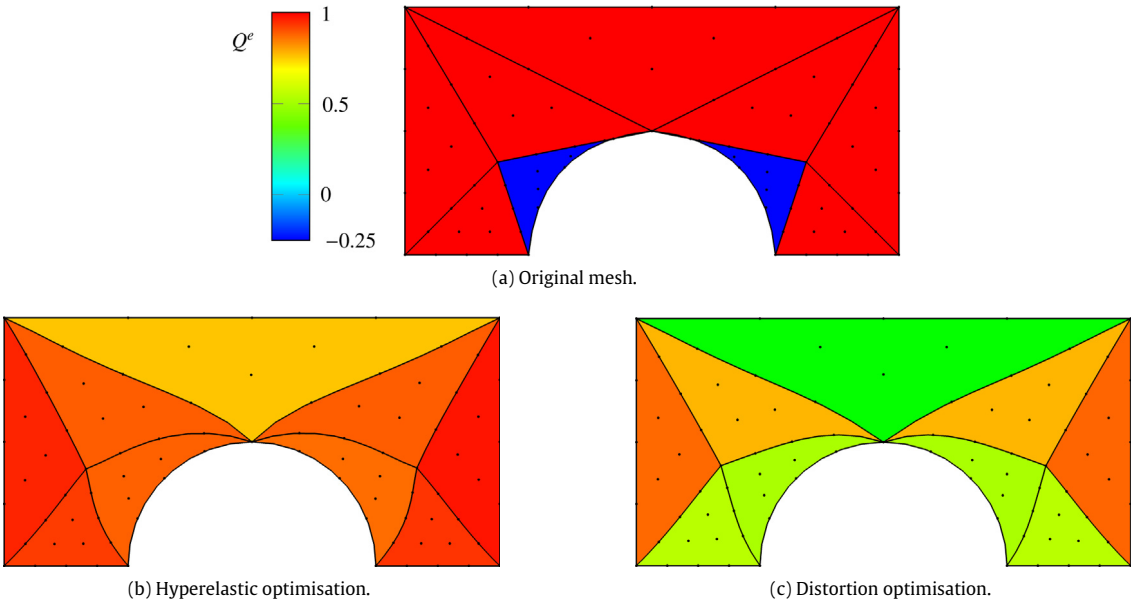
We first discuss Fig. 3a, 3b and 3c, which are all generated by using a very high 40^{th} order integration, namely $Q = 1,681$, using Gauss–Lobatto quadrature rules. This ensures that these terms are all computed to a very high degree of precision. Fig. 3a clearly shows the effect of the Jacobian regularisation discussed in Section 2.2, which causes a rapid increase in the value of the functional as the node approaches the edge of the region of validity. We also see that the functional possesses a minimum at the centre of the element, as is to be expected for this straight-sided quadrilateral. The gradient in Fig. 3b shows large magnitudes in the corners of the validity regions, where the regularisation increases to its largest values, but will clearly drive optimisation towards the centre of the element as intended. Finally, the minimum eigenvalues of the Hessian depicted in Fig. 3c show a smooth profile apart from abrupt maxima in the corners of the limit of validity. Further analysis of the data showed that the vast majority of the region had a very small value, therefore Hessian regularisation would have been used if the optimisation was conducted. We also saw regions of negative eigenvalues in the corners of the element.

Fig. 3d is perhaps the most useful illustration of this analysis process, as it attempts to estimate the amount of over-integration required to accurately calculate the value of the functional. The methodology for this figure is therefore slightly different. We let $\mathcal{E}^q(\nabla\boldsymbol{\phi})$ denote the evaluation of the functional using a polynomial order $q + 2$, and then compute a vector $[\mathcal{E}^0, \ldots, \mathcal{E}^{38}]$, so that the functional is calculated at every polynomial order between 2 and 40. To estimate the 'true' value of the functional, we take the average value of $\mathcal{E}^{34}$ through $\mathcal{E}^{38}$; i.e.

$$\mathcal{E}(\nabla\boldsymbol{\phi}) \approx \frac{1}{5}\sum_{q=34}^{38} \mathcal{E}^q(\nabla\boldsymbol{\phi}).$$

The contour in Fig. 3d then denotes the minimum value of q such that the relative error between $\mathcal{E}$ and $\mathcal{E}^q$ is less than 2%. We observe in the figure that over-integration between $q = 0$ (i.e. integration at order 2) and $q = 10$ (i.e. integration at order 12) is present in this case, with the highest orders of integration required where the element is just outside the region of validity. We posit that locations where larger values of q are required correspond to larger gradient of the functional, where the energy abruptly increases due to the regularisation. Based on the analysis of this figure, we notice that selecting a moderate over-integration of around $q = 6$ is quite

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.



(a) Original mesh.

(b) Hyperelastic optimisation.

(c) Distortion optimisation.

Fig. 4. The optimisation of an initially-invalid two-dimensional example mesh.

acceptable for most cases, providing a good balance between computational efficiency and the accuracy of integration. We therefore select this order of integration for the results in the following section. Overall, we find that this choice is reasonably robust, although in some cases elements that are marginally invalid fail to untangle. We believe that this is due to the error in integration that we see here. Whilst this can be mitigated by using a higher order quadrature, this comes at greatly increased computational cost. The use of an adaptive quadrature could reduce this cost, but we leave this point to future work.

5. Examples of application

This section outlines the application of the variational framework to some two- and three-dimensional examples. We consider the generation of triangular, quadrilateral, tetrahedral and prismatic meshes and combinations thereof. We begin with a brief discussion of how we compare the relative qualities of each . We note that all of the tests below have been performed in parallel using the same 24-core machine, consisting of two Intel(R) Xeon(R) CPU E5-2697v2 processors running at 2.7 GHz. Further information on the meshes presented in this section and their main characteristics, and the performance of the variational algorithm can be found in the table included in Appendix A.

5.1. Quality metric

In order to evaluate the effectiveness of the variational framework, we must determine an element-wise quality measure on which to make our conclusions. A common choice in the literature is the *scaled Jacobian* of the mapping $\boldsymbol{\phi}_M$. It is defined for an element Ω^e as

$$J_s^e = \frac{\min_{\boldsymbol{\xi}\in\Omega_{\text{st}}} J^e_{\boldsymbol{\phi}_M}(\boldsymbol{\xi})}{\max_{\boldsymbol{\xi}\in\Omega_{\text{st}}} J^e_{\boldsymbol{\phi}_M}(\boldsymbol{\xi})} \quad \forall \Omega^e \subset \Omega$$

where $J_{\boldsymbol{\phi}_M} = \det\left(\nabla\boldsymbol{\phi}_M(\boldsymbol{\xi})|_{\Omega^e}\right)$. This definition determines the quality of an element by the ratio of the minimum to maximum determinant of the mapping $\nabla\boldsymbol{\phi}_M$ found within an element. However within the context of this work, this measure has a key drawback: we are optimising the meshes according the mapping $\nabla\boldsymbol{\phi}$, whereas this quality metric is analysing the mapping $\boldsymbol{\phi}_M$. We therefore opt to use the scaled Jacobian of the mapping $\boldsymbol{\phi}$; that is, the definition of J that is used throughout Section 2. The quality element Q^e that we use to analyse meshes is therefore defined to be

$$Q^e = \frac{\min_{\boldsymbol{\xi}}\left[\det(\nabla\boldsymbol{\phi}_M(\boldsymbol{\xi}))\det(\nabla\boldsymbol{\phi}_I^{-1}(\boldsymbol{\xi}))\right]}{\max_{\boldsymbol{\xi}}\left[\det(\nabla\boldsymbol{\phi}_M(\boldsymbol{\xi}))\det(\nabla\boldsymbol{\phi}_I^{-1}(\boldsymbol{\xi}))\right]} \quad \forall \Omega^e \subset \Omega.$$

We further define the overall quality of the mesh by considering the minimum metric over the mesh, defined as

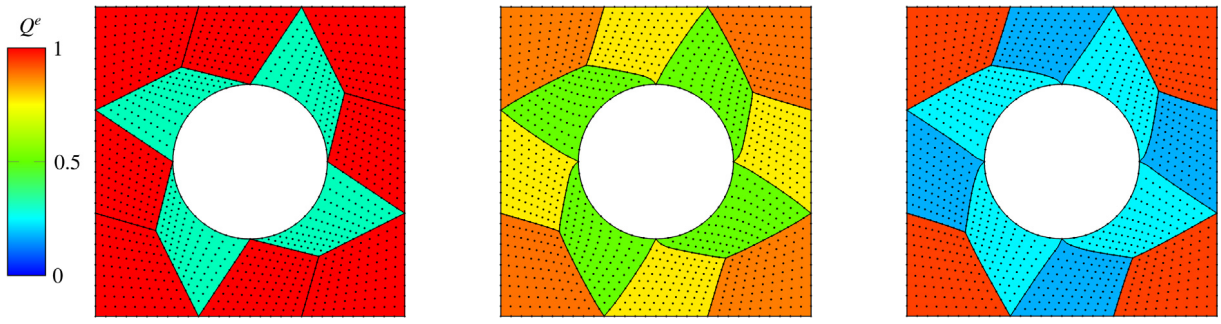
$$Q = \min_{1\le e\le N_{\text{el}}} Q^e.$$

These quality metrics lie in the range $(-\infty, 1]$ and from a physical viewpoint make the assumption that, an 'ideal' element should be as close to straight-sided as possible. Results near $Q^e = 1$ are considered to be the highest quality, as this suggests smoothness of the Jacobian, and any element with $Q^e < 0$ is an invalid element.

The key difference between J_s^e and Q^e is that Q^e provides a measurement of the deformation between the straight-sided and curvilinear element. This makes no difference in the case of triangular and tetrahedral elements (asides from a multiplicative constant) since $\nabla\boldsymbol{\phi}_I$ is a linear mapping. However, in other elements possessing quadrilateral faces, it is possible to have deformation in even a straight sided or planar element due to $\boldsymbol{\phi}_I$ being a quadratic mapping. This new quality metric is therefore general and applicable to any element type, thus allowing us to fairly assess the quality of hybrid meshes.

5.2. Simple two-dimensional demonstration case

Fig. 4 illustrates the intention of the variational framework, which is to introduce domain interior deformation to improve the quality of a curvilinear mesh and correct invalid elements. In this case, the mesh is initially invalid with $Q = -0.24$. This invalidity has been corrected through the regularisation adopted in Section 2.2, and the mesh further improved based on the hyperelastic and distortion functionals. In each of the figures, we visualise the quality distribution Q^e of each element. In this very simple case, an initial mesh of nine triangles, two of which are invalid, are untangled to produce the valid meshes shown in Fig. 4b and 4c showing the ability of the proposed framework to correct invalid



(a) Initial configuration. (b) Optimisation using hyperelastic. (c) Optimisation using distortion.

Fig. 5. Optimisation of 10^{th} order quadrilateral mesh showing the initial configuration and optimisation using the hyperelastic and distortion functionals.

elements. The results demonstrate that elasticity functionals produce a higher quality of the final mesh. Although these are simple meshes, this is an outcome we will see repeated in the forthcoming examples.

5.3. Very high-order mesh

Fig. 5 shows the optimisation of a two-dimensional quadrilateral mesh of 10^{th} order, in order to demonstrate that the framework is still capable of optimising and correcting meshes where the polynomial order is very high. In terms of mesh quality, we observe much the same result as in Fig. 4. In this case, we use an integration of the functional using a 20^{th} order quadrature rule. As noted in the previous section, we find that it is almost always necessary to integrate at a higher order than the mesh itself for the sake of accuracy. In particular, when the mesh is being untangled we find that for the majority of cases it is necessary to integrate at least 6 orders higher than the mesh order. For quadrilateral and hexahedral elements the generation of the suitable quadrature rules at very high orders, as outlined the previous section, is reasonably easy and cheap, and it can be done at runtime for arbitrary orders. However this is not the case for elements with triangular faces, and such quadrature rules have to be pre-computed and stored. The only reason why we do not go higher than fourth order for triangular and tetrahedral meshes in this work is simply we do not currently have suitable integration rules above 10^{th} order. This example of a quadrilateral mesh shows that the method works for very high order approximations in such meshes.

5.4. Cube sphere case

As an example of a three-dimensional geometry, Fig. 6 shows slices through a 4^{th} order tetrahedral mesh of a simple cube geometry, in which a sphere is removed from the centre of the cube. In this the case the mesh begins valid, and the objective of the variational framework is purely improvement of the mesh quality. Again, visually, the meshes optimised by different functionals look very similar and differences can only be drawn when comparing the quality of the elements.

For a more quantitative analysis, we visualise the mesh quality as a function of the global optimisation iteration count in Fig. 7 . On the left we see the residual, measured as the infinity norm of the movement of a node from one iteration to the next. The Winslow and distortion functionals converge to their result much faster than the elasticity methods. The linear elasticity appears to not converge as smoothly as the other functionals, which we believe is due to the order of integration being used. We note that in the definition of the linear elasticity functional, shown in Eq. (2) involves the term $\boldsymbol{E} : \boldsymbol{E} = \|\nabla\boldsymbol{\phi}^\top\nabla\boldsymbol{\phi} - \boldsymbol{I}\|_f^2$. This results in the integration of a polynomial of order $4P$, whereas other functionals only involve the calculation of $\|\nabla\boldsymbol{\phi}\|_f^2$, giving a polynomial of order $2P$. For each of the other functionals, the over-integration we define in Section 4, this appears to be completely sufficient; however the use of $\boldsymbol{E} : \boldsymbol{E}$ means that the linear elasticity requires a higher integration rule. Indeed when operating on smaller 2D meshes we find that the linear elastic functional recovers a smooth convergence profile. This means that in most cases, especially in 3D, the increased cost of using a sufficiently accurate integration for the linear elastic makes it more computationally expensive compared to the other functionals.

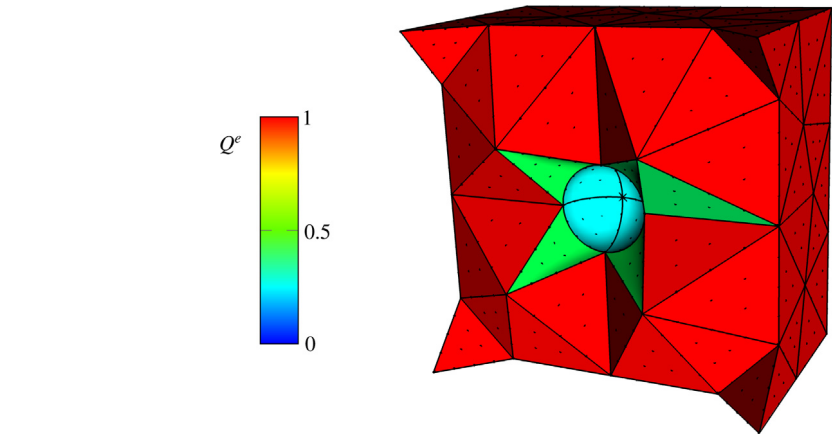
One additional note is that the figure on the right, where we visualise the mesh quality Q as a function of iteration, highlights that very few iterations of the global optimisation are required in order to significantly improve the quality of the mesh. The residual properties are therefore not necessarily that important in the context of mesh optimisation, since because the quality does not improve much beyond the first few iterations. It also establishes the rating of the functionals in terms of the quality of mesh produced: elasticity functionals appear to generate higher quality meshes than the distortion and Winslow functionals.

5.5. Parallel efficiency

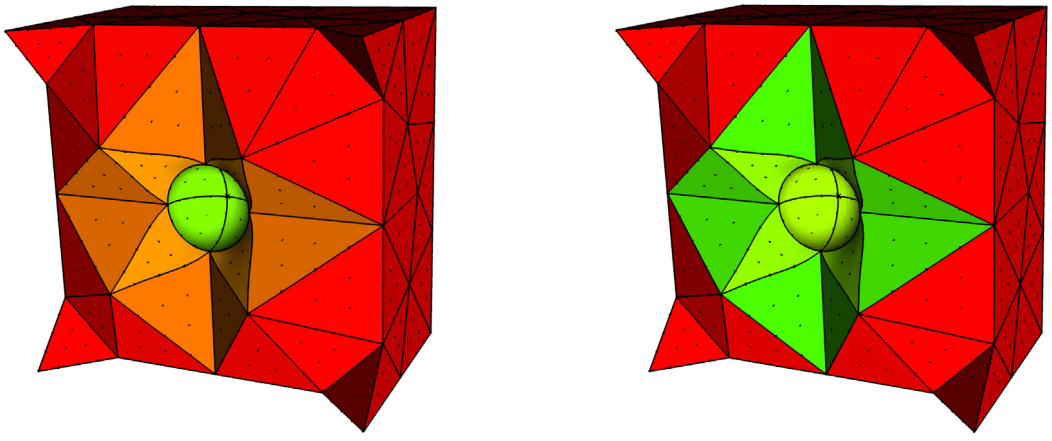
To assess the effectiveness of the simple node colouring parallelisation strategy outlined in algorithm 2 , we first show the results of a strong scaling simulation, wherein the mesh remains unchanged between simulations whilst increasing the number of threads from 1 to 24 that run in the parallel execution phase of the algorithm. A simple test mesh of a sphere inside a cube has been constructed, the geometry of which can be seen in Fig. 6. However we use a far denser mesh in this example, constructing an initial mesh containing 32 928 tetrahedra at polynomial order $P = 4$. This yields an optimisation problem with approximately 1 million degrees of freedom. We discard setup costs and node colouring, since this portion of the algorithm is not parallelised, and examine the time taken to perform 10 iterations of the global optimisation procedure. Utilising the hyperelastic functional, this results in a mesh quality increase from $Q = 0.2$ to $Q = 0.65$,

The timings for this can be seen in Fig. 8, where the ideal scaling is visualised as a black line. We observe an scaling of 70% efficiency between 1 and 24 cores (runtimes 326.5 s and 20.0 s respectively), which is excellent given the size of the problem and the simple node colouring strategy used. It is likely that this can be improved further using a more optimal colouring strategy. However for this work the efficiency has substantially reduced the runtime of the optimisation process in an effective manner.

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.



(a) Original configuration.



(b) Optimisation using the hyperelastic analogy.

(c) Optimisation using the distortion metric.

Fig. 6. Optimisation of 4th order sphere mesh from the initial configuration using the hyperelastic and distortion functionals.

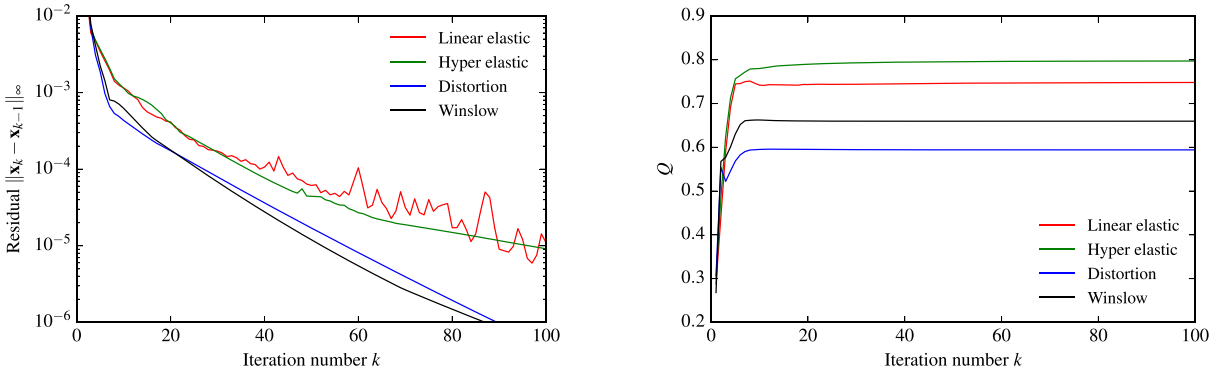


Fig. 7. Shows the displacement residual and quality, Q, of the cube sphere mesh for each functional.

5.6. DLR F6

We now consider a more complex geometry in order to demonstrate the effectiveness of the framework on test cases of interest to the aeronautics industry, by examining a mesh of a DLR F6 geometry. The initial coarse mesh, which comprises of approximately 100,000 tetrahedra, can be seen in the top left position of Fig. 9, where we show the elements that are of quality $Q^e < 0.5$. This case also has a number of invalid elements, which can be seen in the top right figure. This inset shows the distribution of Q^e, in which a number of invalid elements can be seen having quality less than 0. We again use the hyperelastic functional to both untangle and optimise the mesh, which results in a significant overall improvement in element quality. The resulting distribution is seen in the bottom right figure, where a noticeable shift towards $Q^e = 1$ can be observed. A few very poor quality tetrahedra remain, which can be seen in the bottom left of the figure. These are solely due to the initial linear mesh, which in this region contains a number of flat elements, which in turn limits the capability of the framework. This highlights need for further improvement in

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.

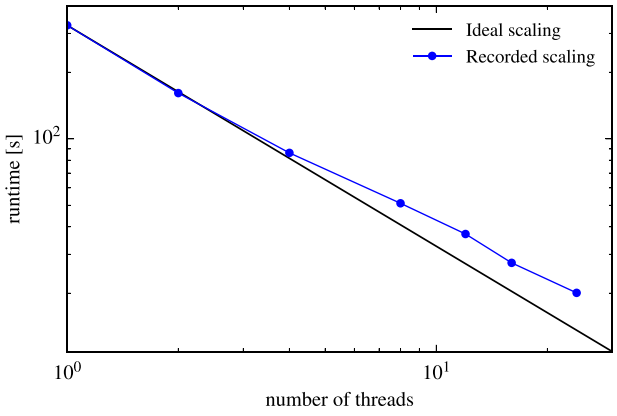


Fig. 8. Parallel scalability of a sphere test case consisting of 32,928 tetrahedra with 324,364 free nodes at $P = 4$.

linear mesh generation for high-order generation. An interesting additional point to note is that the hyperelastic functional was the only one able to untangle the mesh from the initial invalid configuration. We posit that further investigation into the integration order is required to further understand this phenomenon.

5.7. CAD sliding

Fig. 10 highlights the effects of the CAD sliding outlined in Section 5.7. In this example, we take a flat surface and place a semi-sphere onto it. This generates an initial curved mesh, visualised on the left hand side, which possesses 8 invalid elements. Taking a closer look at the initial mesh, it is very clear that the surface mesh induces an invalidity where the sphere meets the flat plane. The ability to slide the element edges along the flat plane and additionally the surface of the sphere is therefore required in order to have any chance of generating a valid mesh. The optimised mesh on the right-hand side shows how the deformation is incorporated into the surface edges, deforming them appropriately in order to produce a valid and very high-quality mesh, as can be see from the quality metric.

We note that, other than the small expense of querying the surface metric via the CAD interface, the computational cost of optimising the position of nodes on the surface is not significantly higher than the cost moving the interior nodes within the domain. Combined with the fact that surface nodes represent a small proportion of the nodes to be optimised, we find that the use of CAD sliding *does not* cause a disproportionate increase in computational expense.

5.8. Boeing reduced landing gear

In our final example, we show results for optimisation of another well-known complex geometric example: the Boeing reduced landing gear. In this case, we have created a hybrid mesh containing a prismatic boundary layer, filled with tetrahedra in the interior. The purpose of the prism layers is to capture the wall-normal flow physics, where very large gradients of the flow velocities occur close to the surface. Since this region contains very high shear, the prismatic elements should substantially decrease in thickness near the wall so that they become highly stretched

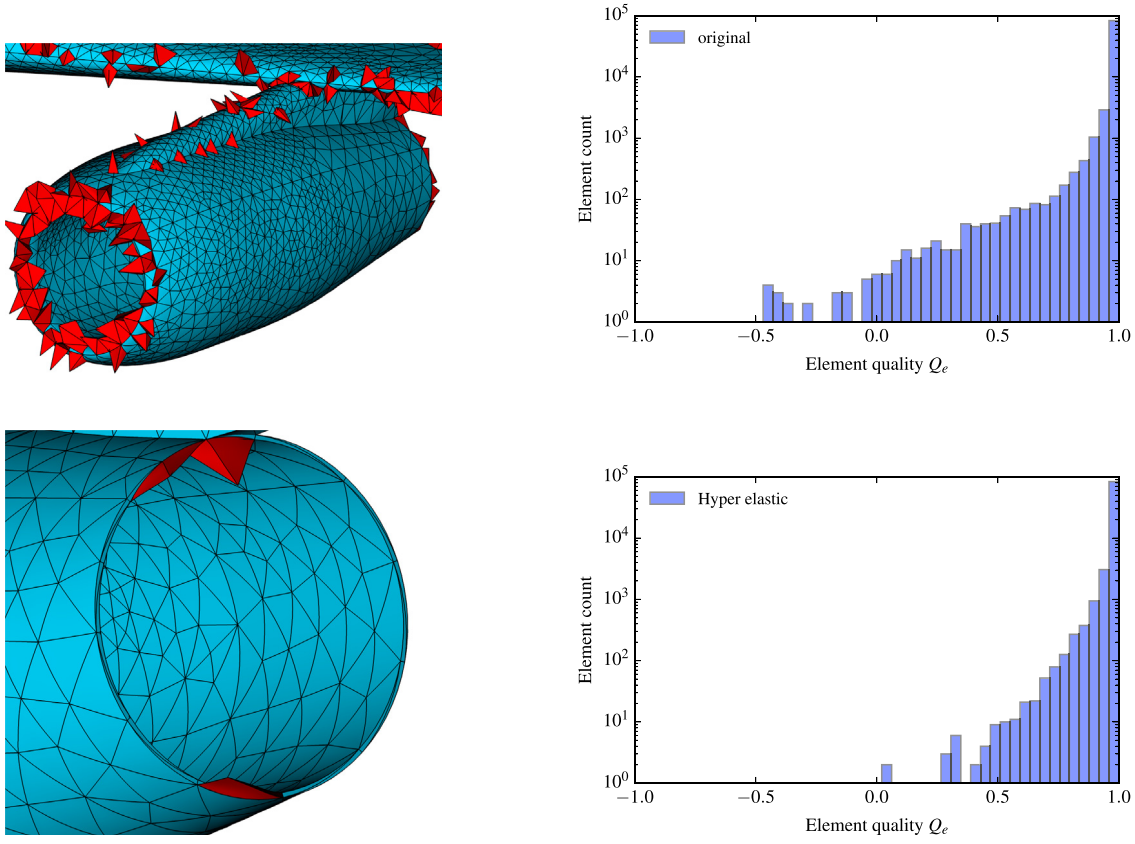


Fig. 9. Untangling and optimisation of the DLR F6 geometry. The left figures show the mesh before and after optimisation. On the right we show the distribution of the quality metric Q^e before and after optimisation with the hyperelastic functional.

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.

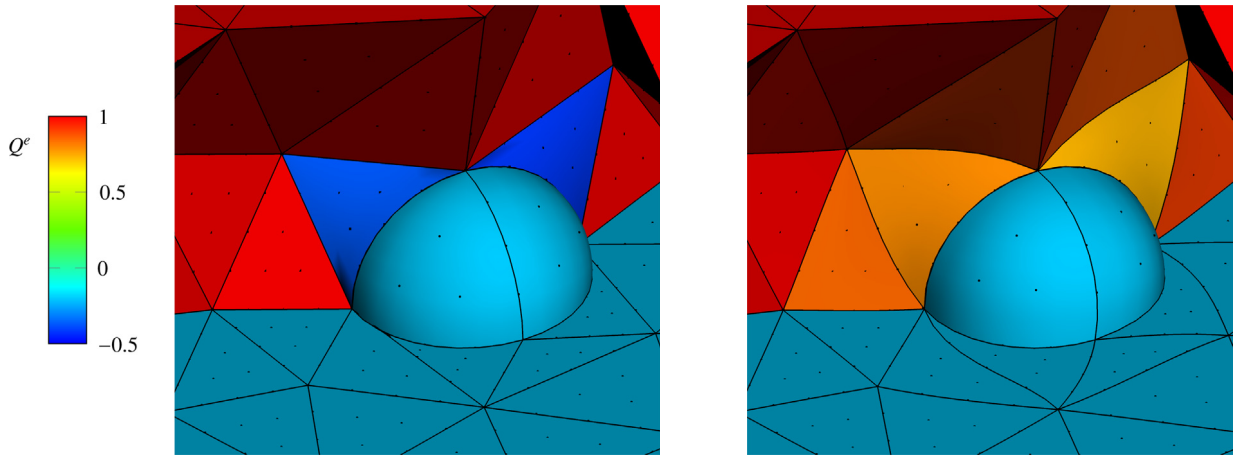


Fig. 10. Cross-section of a semi-sphere case highlighting the sliding of CAD curves along the surface. The left-hand image shows the initial mesh and the right-hand figure shows the optimised mesh. Note that the colour of the surface triangles is not related to mesh quality.

relative to the tangential surface direction. This poses a substantial challenge for curved boundary layer generation. If we apply curvature to a standard linear boundary layer mesh, it is all but guaranteed for all but the most simple geometries that there will be a large number of invalid elements. Given that there is very little space available to accommodate the curvature of the boundary, correcting these boundary layer elements becomes very difficult. Further, the number of elements to optimise also increases substantially, thus increasing the computational cost of the method.

As has been noted in previous work it is far more practical and robust for high-order meshing to generate a single 'macro' isotropic prism at the geometric boundary, in which the curvature of the surface can be readily applied, and then use a method of isoparametric splitting to produce the anisotropic elements [41]. Adopting this approach here, we first generate a linear hybrid mesh combining tetrahedral elements and triangular prismatic 'macro' elements, introduce the boundary curvature and then apply the variational optimisation to optimise the quality of the mesh. We then finish the mesh by applying isoparametric splitting to obtain the desired boundary-layer thickness.

Fig. 11 shows the 'macro' mesh before and after optimisation, for which we have used the hyperelastic functional since this has been shown to produce the highest quality meshes. We also depict the final mesh created after the macro layer has been split. For the purposes of clarity, the tetrahedra have been removed. Overall the figure illustrates that whilst the initial configuration before optimisation is of a reasonable quality, there are a number of lower-quality elements on the shoulders of the tyres. The quality in this area, as well as throughout the mesh generally, is then improved in optimisation across all of the elements shown. The figure also provides evidence that this approach produces a high quality mesh after splitting the prismatic layers.

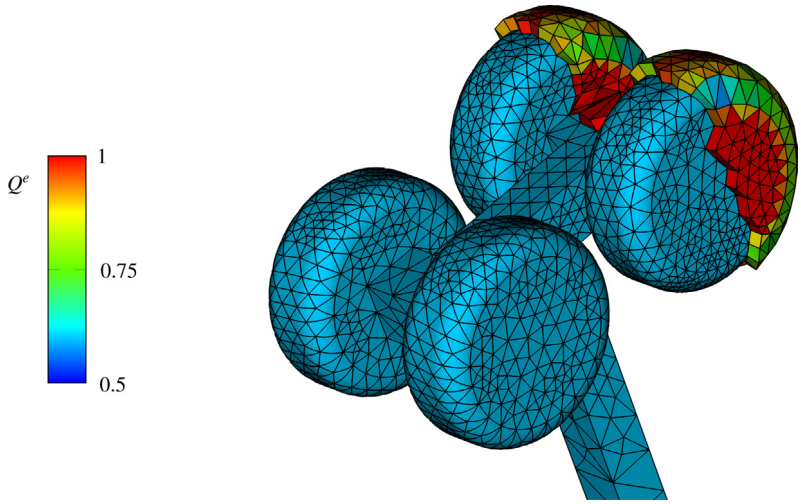
To quantify the increase in element quality, we show a number of element quality histograms for this case in Fig. 12. Firstly, the overall distribution from the initial configuration seen in Fig. 12a improves substantially under optimisation, as shown in Fig. 12b, where we can observe a clear shift to the right. However, we note that this optimisation was conducted with a material constant of $\nu = 0.45$, which means that the elastic solid which is being relaxed is very stiff. Fig. 12c shows that reducing ν to 0.4 leads to a mesh that, whilst being improved over the initial configuration, is overall of a lower quality compared to $\nu = 0.45$. This observation aligns well with the results reported in Ref. [6], where meshes generated using values of ν close to the incompressibility limit lead to higher quality elements. Curiously, we observe that both the distortion and Winslow functionals lead to a decreased quality of the mesh, as shown in Fig. 12d for the distortion functional.

We stress that the use of the isoparametric splitting of the macro layer is a necessity when generating high-order anisotropic boundary layers. We have found in practice that, when using the variational framework on anisotropic elements, the optimisation algorithm would very regularly fail to find a new minimum. The reason for this behaviour is that the sensitivity of the functional to nodal location is very strong due to the highly stretched shape of the prismatic elements, and this leads to the very large gradients that we observe in the optimisation process. As discussed in Section 4, when the gradient is large, high-degree quadrature is required and the $Q = P + 6$ rule is not sufficient. While an adaptive or very high-order integration would allow for the optimisation of anisotropic elements, it is simply not required when using the isoparametric splitting, meaning that this approach is computationally more viable. While other examples of *a posteriori* high-order mesh generation have shown the ability to correct anisotropic elements, they regularly report the need for greatly increased iteration count in the solution of either their PDEs or optimisation processes. We therefore postulate that this is counterpart of what we see with failed optimisation.

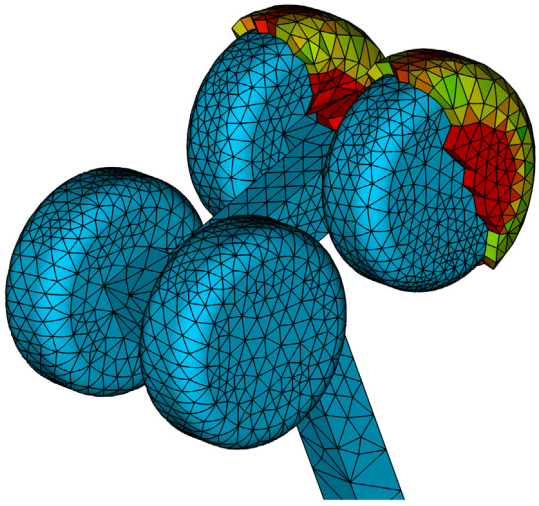
6. Conclusions

We have presented a parallel implementation of a variational framework for curvilinear mesh generation through the optimisation of valid and correction of invalid meshes. We have formulated the problem of generating a high-order mesh as that of finding a mapping from a straight-sided mesh to a curvilinear mesh that conforms to the boundary of the domain. We recast the problem of finding this mapping as a variational problem where the curvilinear mesh is obtained as the minimum of a functional with nodal mesh displacements constrained at the boundary. Interpretation of the functional as a deformation energy permits its analysis through the theory of solid mechanics and take advantage of the physical insights and theoretical results provided by that theory. Of particular interest here are the conditions that a functional must satisfy to ensure the existence of a minimum and the various constitutive equations proposed for the energy functional to model a variety of material deformations. However, it should be stressed that the functionals for mesh generation do not necessarily have to be physically meaningful. For instance, during the untangling of

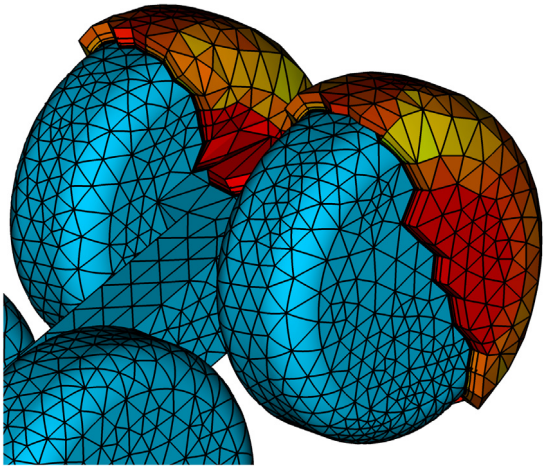
Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.



(a) Pre-optimisation.



(b) Post-optimisation.



(c) After isoparametric splitting.

Fig. 11. Hybrid prismatic–tetrahedral mesh of the Boeing reduced landing gear configuration before (a) and after (b) optimisation, and after the isoparametric splitting is applied (c). Note that the colour of the surface triangles is not related to mesh quality. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

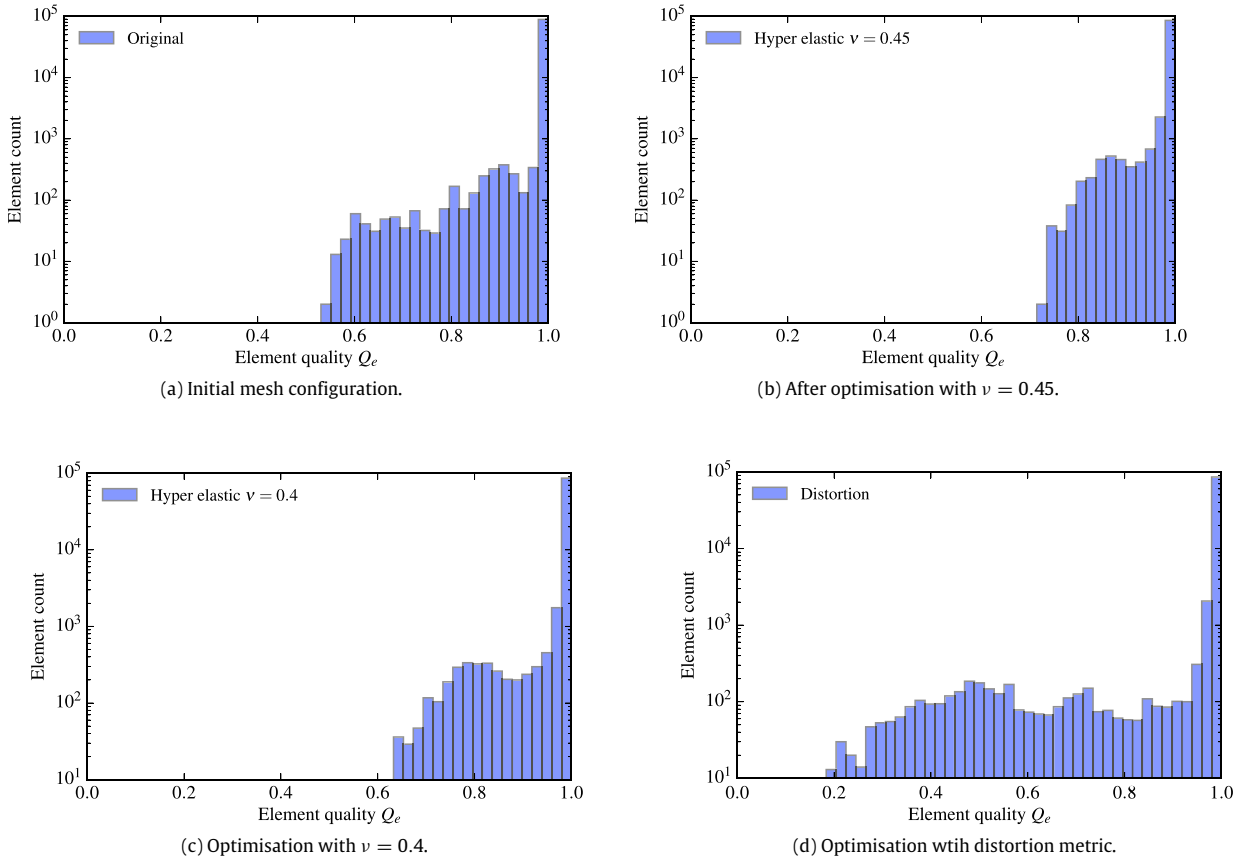
invalid meshes, the Jacobian could be negative thus violating the principle of non-interpenetrability of matter.

The variational formulation provides a general framework for finding the mapping that offers a number of benefits. We have been able to devise energy functionals for the majority of mapping-based high-order mesh generation methods proposed in the literature to date. Further, we have shown the equivalence of the Winslow method with the minimisation of the shape distortion measure in two dimensions. Numerical experiments also show their behaviour and performance to be very similar in three-dimensional cases. This framework permits a modular implementation of these methods since implementing a new one only requires to change modules defining the energy functional and its derivatives. It has also allowed us to perform comparisons of the performance of the various methods. We have found that elasticity methods produce better quality meshes, but require more iterations to converge than the Winslow or shape distortion methods.

A further contribution of the paper is the development of a very efficient Hessian-based optimisation algorithm, using a relaxation technique during the iteration that, through a partitioning of the domain via colouring, lends itself to a parallel implementation with excellent scalability. We have observed 70% efficiency between 1 and 24 2.7 GHz cores in a mesh with 324,364 nodes that could be optimised in just 20 s. The optimisation algorithm has also been extended with the inclusion of Jacobian and Hessian regularisation to deal with tangled meshes with negative Jacobians. Finally, we allow nodes to move, or "slide", on the curves and surfaces which represent the boundary of the domain. These curves and surfaces are represented by CAD entities defined in their respective parametric spaces. This requires the expressions of the gradients and Hessians in the optimisation process to be evaluated with respect to the parametric coordinates. The ability of the method to deal with nodes sliding on the CAD entities was illustrated using some practical examples. Finally, the examples presented in the results section have demonstrated that the method is able to deal with complex geometries and optimise very sizeable meshes very efficiently.

This framework is sufficiently general and flexible to incorporate element quality control by modifying the functional to be optimised. We envisage that this can be done either by incorporating thermal stress terms, as proposed in previous work [7], within the energy functional or, alternatively, through the use of monitor functions, as advocated in Ref. [16]. These modified versions of the energy functional could be used for the generation of anisotropic

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.



(a) Initial mesh configuration.

(b) After optimisation with $\nu = 0.45$.

(c) Optimisation with $\nu = 0.4$.

(d) Optimisation wtih distortion metric.

Fig. 12. Element quality histograms of the Boeing reduced landing gear configuration for initial configuration and various optimisation settings.

meshes where the solution field is highly stretched as, for instance, in boundary layers and wakes.

Acknowledgements

MT acknowledges Airbus and EPSRC for funding under an industrial CASE studentship. DM acknowledges support from the EU Horizon 2020 project ExaFLOW (grant 671571) and the PRISM project under EPSRC grant EP/L000407/1. JP would like to acknowledge discussions with Prof. Javier Bonet on the problem of existence in hyperelasticity that led us to this direction of research in mesh generation. We also thank Jan Eichstädt for assistance in proofreading and validating the numerical code underpinning this work.

Appendix A. Algorithmic performance

Table A.1 contains details of the meshes employed, the main algorithmic parameters, and the performance of the proposed variational method for the examples presented in Section 5.

Appendix B. Analytic derivatives

We derive analytic expressions for the energy functionals in this appendix. We recall that the following nomenclature, depicted in Fig. 1, is used in Section 2:

- The standard element is Ω_{st} with coordinates $\boldsymbol{\xi} \in \Omega_{\text{st}}$. This has N nodal points which we denote by $\boldsymbol{\xi}^i$.
- The curvilinear element is Ω^e with coordinates $\boldsymbol{x} \in \Omega^e$. The nodes of the element are $\boldsymbol{x}^i$.
- The ideal element is Ω_I^e with coordinates $\boldsymbol{y} \in \Omega_I^e$ and nodes $\boldsymbol{y}^i$.

We then consider the mappings:

- $\boldsymbol{\phi}_M : \Omega_{\text{st}} \to \Omega^e$ which maps the reference element to the curvilinear element;
- $\boldsymbol{\phi}_I : \Omega_{\text{st}} \to \Omega_I^e$ which maps the reference element to the ideal element; and
- $\boldsymbol{\phi} : \Omega_I^e \to \Omega^e$ which maps the ideal element to the curvilinear element, and can be viewed as the composition $\boldsymbol{\phi}_M \circ \boldsymbol{\phi}_I^{-1}$.

Each of these mappings is defined in the isoparametric sense, so that for example

$$\boldsymbol{x} = \boldsymbol{\phi}_M(\boldsymbol{\xi}) = \sum_{n=1}^{N} \boldsymbol{x}^n \ell_n(\boldsymbol{\xi}),$$

where $\ell_n(\boldsymbol{\xi})$ has the Lagrange interpolant property that $\ell_n(\boldsymbol{\xi}^m) = \delta_{nm}$, with δ_{nm} being the Kronecker delta. We then have that $\boldsymbol{\phi}(\boldsymbol{\xi}^i) = \boldsymbol{x}^i$; i.e. each reference nodal point $\boldsymbol{\xi}^i$ maps onto an associated curvilinear point $\boldsymbol{x}^i$ in an isoparametric fashion.

When running the local optimisation procedure of algorithm 1 , we select a node, say $\boldsymbol{x}^n$, which is connected to one or more elements that neighbour the node. We then wish to evaluate the functional

$$\mathcal{E}(\nabla\boldsymbol{\phi}) = \int_{\Omega_I^e} W(\nabla\boldsymbol{\phi}(\boldsymbol{y}))\,\mathrm{d}\boldsymbol{y}$$

on a subset of elements that are connected via some node of the mesh, where $\nabla\boldsymbol{\phi}$ denotes the Jacobian matrix

$$\nabla\boldsymbol{\phi}(\boldsymbol{y}) = \nabla\boldsymbol{\phi}_M(\boldsymbol{\phi}_I^{-1}(\boldsymbol{y}))\nabla\boldsymbol{\phi}_I^{-1}(\boldsymbol{y}).$$

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.

Table A.1
Summary of relevant statistics from simulations in Section 5: polynomial order P, integration order Q, degrees of freedom in the mesh, number of threads and iterations taken for convergence, scaled Jacobian before and after simulation and the runtime for the optimisation procedure.

Case	P	Q	N_{el}	Degrees of freedom	Number of threads	Number of iterations	min Q_e (start)	min Q_e (end)	Time [s]
Fig. 4	4	10	9	118	1	25	−0.45	0.72	0.11
Fig. 5	10	20	12	2,240	24	25	0.31	0.57	4.79
Fig. 6	4	10	276	6,810	24	25	0.32	0.79	4.69
Fig. 8	4	10	32,928	973,092	24	10	0.23	0.65	20.00
Fig. 9	4	10	88,545	2,673,057	24	5	−5.77	0.05	152.35
Fig. 10	4	10	294	10,061	24	25	−1.12	0.73	17.23
Fig. 11	4	10	91,545	3,089,019	24	10	0.53	0.58	345.62

For the numerical integration of the functional, we will select evaluation points $\tilde{\boldsymbol{\xi}}^q$ and associated quadrature weights on Ω_{st}, and then map the integral back to there, so that

$$\mathcal{E}(\nabla\boldsymbol{\phi}) = \int_{\Omega_{\text{st}}} W\left[\nabla\boldsymbol{\phi}_M(\boldsymbol{\xi})\nabla\boldsymbol{\phi}_I^{-1}(\boldsymbol{\phi}_I(\boldsymbol{\xi}))\right]\det(\nabla\boldsymbol{\phi}_I)\,\mathrm{d}\boldsymbol{\xi}.$$

The optimisation requires both the gradients and the Hessian of $\mathcal{E}$ with respect to the positions of $\boldsymbol{x}^n$, i.e.

$$\frac{\partial}{\partial x_m^n}\mathcal{E}(\nabla\boldsymbol{\phi}) = \int_{\Omega_{\text{st}}} \frac{\partial}{\partial x_m^n}\left\{W\left[\nabla\boldsymbol{\phi}_M(\boldsymbol{\xi})\nabla\boldsymbol{\phi}_I^{-1}(\boldsymbol{\phi}_I(\boldsymbol{\xi}))\right]\det\boldsymbol{\phi}_I(\boldsymbol{\xi})\right\}\mathrm{d}\boldsymbol{\xi},$$

where $\boldsymbol{x}^n = (x_1^n, x_2^n, x_3^n)$ for a three-dimensional simulation. We then apply the product rule to see that

$$\begin{aligned}\frac{\partial}{\partial x_m^n}\mathcal{E}(\nabla\boldsymbol{\phi}) = &\int_{\Omega_{\text{st}}} \frac{\partial}{\partial x_m^n}\left\{W\left[\nabla\boldsymbol{\phi}_M(\boldsymbol{\xi})\nabla\boldsymbol{\phi}_I^{-1}(\boldsymbol{\phi}_I(\boldsymbol{\xi}))\right]\right\}\det\boldsymbol{\phi}_I(\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi}\\ &+\int_{\Omega_{\text{st}}} W\left[\nabla\boldsymbol{\phi}_M(\boldsymbol{\xi})\nabla\boldsymbol{\phi}_I^{-1}(\boldsymbol{\phi}_I(\boldsymbol{\xi}))\right]\frac{\partial}{\partial x_m^n}\left[\det\boldsymbol{\phi}_I(\boldsymbol{\xi})\right]\mathrm{d}\boldsymbol{\xi}.\end{aligned}$$

Since the original linear mesh remains fixed and thus the ideal mapping $\boldsymbol{\phi}_I$ remains unchanged under node displacement, there is no dependence on the Cartesian coordinates x_m^n. The derivative of $\det\boldsymbol{\phi}_I(\boldsymbol{\xi})$ is therefore zero, meaning that the second integral can be ignored.

To evaluate the first integral however, we clearly need to consider the various forms of W. We do note that each of the functionals considered in this article are some combination of the terms $\|\nabla\boldsymbol{\phi}_M\|_f^2$ and $J_R(\nabla\boldsymbol{\phi}_M)$, where $J_R(\nabla\boldsymbol{\phi}_M)$ is the regularised Jacobian used to untangle elements, and the points of evaluation have been omitted for clarity. Obtaining the derivatives of these terms is accomplished as follows.

We first rewrite the Frobenius norm $\|\nabla\boldsymbol{\phi}\|_f^2 = \nabla\boldsymbol{\phi} : \nabla\boldsymbol{\phi}$, and use the matrix identity

$$\frac{\partial}{\partial x}(\boldsymbol{A} : \boldsymbol{A}) = 2\frac{\partial\boldsymbol{A}}{\partial x} : \boldsymbol{A}$$

to obtain

$$\frac{\partial}{\partial x_m^n}\|\nabla\boldsymbol{\phi}\|_f^2 = 2\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1} : \nabla\boldsymbol{\phi}_M\nabla\boldsymbol{\phi}_I^{-1}. \tag{B.1}$$

Now, using the definition of the regularised Jacobian from Eq. (5) with $J = \det\nabla\boldsymbol{\phi}_M(\boldsymbol{\xi})$, we apply the chain rule to obtain

$$\frac{\partial J_R}{\partial x_m^n} = \frac{1}{2}\frac{\partial J}{\partial x_m^n}\left(1 + \frac{J}{\sqrt{4\delta^2 + J^2}}\right) = \frac{1}{2}\frac{\partial J}{\partial x_m^n}\left(1 + \frac{J}{2J_R - J}\right). \tag{B.2}$$

Both Eqs. (B.1) and (B.2) lead to terms involving the derivative of $\nabla\boldsymbol{\phi}_M$ and its determinant. Following a similar approach to [42] and noting that we use an isoparametric projection, we have that

$$\boldsymbol{x} = \boldsymbol{\phi}_M(\boldsymbol{\xi}) = \sum_{k=1}^{N}\boldsymbol{x}^k\ell_k(\boldsymbol{\xi}) \quad\Rightarrow\quad [\nabla\boldsymbol{\phi}_M(\boldsymbol{\xi})]_{ij} = \sum_{k=1}^{N} x_i^k\left[\frac{\partial\ell_k}{\partial\xi_j}(\boldsymbol{\xi})\right]_i.$$

The derivative is therefore given by

$$\left[\frac{\partial}{\partial x_m^n}\nabla\boldsymbol{\phi}_M(\boldsymbol{\xi})\right]_{ij} = \sum_{k=1}^{N}\delta_{nk}\delta_{im}\left[\frac{\partial\ell_k}{\partial\xi_j}(\boldsymbol{\xi})\right]_i = \delta_{im}\left[\frac{\partial\ell_k}{\partial\xi_j}(\boldsymbol{\xi})\right]_i.$$

We then require the derivative of J to finalise the derivative of J_R. To achieve this we can use the matrix identity

$$\frac{\partial}{\partial x}\det(\boldsymbol{A}) = \det(\boldsymbol{A})\,\mathrm{tr}\left[\boldsymbol{A}^{-1}\frac{\partial\boldsymbol{A}}{\partial x}\right],$$

which then gives us the derivative of J in terms of the calculated quantities above. With these expressions, we can compute first- and second-order derivatives for each of the functionals by using standard product, chain and quotient rules. Below, we outline the analytic first and second derivatives for the four functionals used in this article, which have been verified by using finite difference approximations.

Non-linear elasticity

$$\begin{aligned}W &= \frac{\mu}{2}\left(\|\nabla\boldsymbol{\phi}\|_f^2 - 3\right) - \mu\ln J_R + \frac{\kappa}{2}(\ln J_R)^2\\ \frac{\partial}{\partial x_m^n}W &= \mu\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1} : \nabla\boldsymbol{\phi}\right) + \frac{\partial J}{\partial x_m^n}\frac{1}{2J_R - J}(\kappa\ln J_R - \mu)\\ \frac{\partial}{\partial x_m^n x_p^n}W &= \mu\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1} : \frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_p^n}\nabla\boldsymbol{\phi}_I^{-1}\right)\\ &\quad + \frac{\partial J}{\partial x_m^n}\frac{\partial J}{\partial x_p^n}\frac{1}{(2J_R - J)^2}\left(\kappa - J\frac{\kappa\ln J_R - \mu}{2J_R - J}\right).\end{aligned}$$

Linear elasticity

$$\begin{aligned}W &= \frac{\kappa}{2}(\ln J_R) + \mu\left\|\frac{1}{2}(\nabla\boldsymbol{\phi}^\top\nabla\boldsymbol{\phi} - \boldsymbol{I})\right\|_f^2\\ \frac{\partial}{\partial x_m^n}W &= 2\mu\left[\frac{1}{2}\left((\nabla\boldsymbol{\phi}_I^{-1})^\top\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\right)^\top\nabla\boldsymbol{\phi}_M\nabla\boldsymbol{\phi}_I^{-1}\right.\right.\\ &\quad\left.+(\nabla\boldsymbol{\phi}_I^{-1})^\top(\nabla\boldsymbol{\phi}_M)^\top\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1}\right)\\ &\quad\left.\times : \frac{1}{2}\left((\nabla\boldsymbol{\phi}_I^{-1})^\top(\nabla\boldsymbol{\phi}_M)^\top\nabla\boldsymbol{\phi}_M\nabla\boldsymbol{\phi}_I^{-1} - \boldsymbol{I}\right)\right]\\ &\quad + \frac{\kappa\ln J_R}{2J_R - J}\frac{\partial J}{\partial x_m^n}\\ \frac{\partial}{\partial x_m^n x_p^n}W &= 2\mu\left[\left((\nabla\boldsymbol{\phi}_I^{-1})^\top\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\right)^\top\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_p^n}\nabla\boldsymbol{\phi}_I^{-1}\right)\right.\end{aligned}$$

$$\times : \frac{1}{2}\left((\nabla\boldsymbol{\phi}_I^{-1})^\top(\nabla\boldsymbol{\phi}_M)^\top\nabla\boldsymbol{\phi}_M\nabla\boldsymbol{\phi}_I^{-1} - \boldsymbol{I}\right)\Bigg]$$

$$+2\mu\Bigg[\frac{1}{2}\Bigg((\nabla\boldsymbol{\phi}_I^{-1})^\top\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\right)^\top\nabla\boldsymbol{\phi}_M\nabla\boldsymbol{\phi}_I^{-1} + (\nabla\boldsymbol{\phi}_I^{-1})^\top(\nabla\boldsymbol{\phi}_M)^\top\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1}\Bigg) : \frac{1}{2}\Bigg((\nabla\boldsymbol{\phi}_I^{-1})^\top\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_p^n}\right)^\top\nabla\boldsymbol{\phi}_M\nabla\boldsymbol{\phi}_I^{-1} + (\nabla\boldsymbol{\phi}_I^{-1})^\top(\nabla\boldsymbol{\phi}_M)^\top\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_p^n}\nabla\boldsymbol{\phi}_I^{-1}\Bigg)\Bigg]$$

$$+\frac{\kappa}{(2J_R - J)^2}\frac{\partial J}{\partial x_m^n}\frac{\partial J}{\partial x_p^n}\left(1 - \frac{J\ln J_R}{2J_R - J}\right).$$

Distortion

$$W = \frac{\|\nabla\boldsymbol{\phi}\|_f^2}{n|J_R|^{2/n}}$$

$$\frac{\partial}{\partial x_m^n}W = 2W\left[\frac{\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1} : \nabla\boldsymbol{\phi}\right)}{(\nabla\boldsymbol{\phi} : \nabla\boldsymbol{\phi})} - \frac{\partial J}{\partial x_m^n}\frac{1}{n(2J_R - J)}\right]$$

$$\frac{\partial}{\partial x_m^n x_p^n}W = \frac{1}{W}\frac{\partial W}{\partial x_m^n}\frac{\partial W}{\partial x_p^n} + 2W\Bigg[\frac{\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1} : \frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1}\right)}{(\nabla\boldsymbol{\phi} : \nabla\boldsymbol{\phi})} - 2\frac{\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1} : \nabla\boldsymbol{\phi}\right)\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1} : \nabla\boldsymbol{\phi}\right)}{(\nabla\boldsymbol{\phi} : \nabla\boldsymbol{\phi})^2} + \frac{\partial J}{\partial x_m^n}\frac{\partial J}{\partial x_p^n}\frac{J}{n(2J_R - J)^3}\Bigg].$$

Winslow

$$W = \frac{\|\nabla\boldsymbol{\phi}\|_f^2}{J_R}$$

$$\frac{\partial}{\partial x_m^n}W = W\left[\frac{2\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1} : \nabla\boldsymbol{\phi}\right)}{(\nabla\boldsymbol{\phi} : \nabla\boldsymbol{\phi})} - \frac{\partial J}{\partial x_m^n}\frac{1}{(2J_R - J)}\right]$$

$$\frac{\partial}{\partial x_m^n x_p^n}W = \frac{1}{W}\frac{\partial W}{\partial x_m^n}\frac{\partial W}{\partial x_p^n} + 2W\Bigg[\frac{\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1} : \frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1}\right)}{(\nabla\boldsymbol{\phi} : \nabla\boldsymbol{\phi})} - 2\frac{\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1} : \nabla\boldsymbol{\phi}\right)\left(\frac{\partial\nabla\boldsymbol{\phi}_M}{\partial x_m^n}\nabla\boldsymbol{\phi}_I^{-1} : \nabla\boldsymbol{\phi}\right)}{(\nabla\boldsymbol{\phi} : \nabla\boldsymbol{\phi})^2} + \frac{\partial J}{\partial x_m^n}\frac{\partial J}{\partial x_p^n}\frac{J}{2(2J_R - J)^3}\Bigg].$$

References

[1] Vincent P, Jameson A. Facilitating the adoption of unstructured high-order methods amongst a wider community of fluid dynamicists. Math Model Nat Phenom 2011;6(3):97–140.
[2] Wang ZJ, Fidkowski K, Abgrall R, Bassi F, Caraeni D, Cary A, et al. High-order CFD methods: Current status and perspective. Internat J Numer Methods Fluids 2013;72(8):811–45.
[3] Xie Z, Sevilla R, Hassan O, Morgan K. The generation of arbitrary order curved meshes for 3D finite element analysis. Comput Mech 2013;51:361–74.
[4] Hartmann R, Leicht T. Generation of unstructured curvilinear grids and high-order discontinuous Galerkin discretization applied to a 3D high-lift configuration. Internat J Numer Methods Fluids 2016;82:316–33.
[5] Persson P-O, Peraire J. Curved mesh generation and mesh refinement using Lagrangian solid mechanics. In: 47th AIAA aerospace sciences meeting and exhibit, Orlando (FL), USA, AIAA paper 2009–949. 2009.
[6] Poya R, Sevilla R, Gil AJ. A unified approach for a posteriori high-order curved mesh generation using solid mechanics. Comput Mech 2016;58(3):457–90.
[7] Moxey D, Ekelschot D, Keskin Ü, Sherwin S, Peiró J. A thermo-elastic analogy for high-order curvilinear meshing with control of mesh validity and quality. Procedia Eng 2014;82:127–35.
[8] Fortunato M, Persson P-O. High-order unstructured curved mesh generation using the Winslow equations. J Comput Phys 2016;307:1–14.
[9] Dey S, O'Bara R, Shephard M. Curvilinear mesh generation in 3D. In: Proceedings of the 8th international meshing roundtable, South Lake Tahoe, California. 1999.
[10] Sherwin S, Peiró J. Mesh generation in curvilinear domains using high-order elements. Internat J Numer Methods Engrg 2002;53:207–23.
[11] Toulorge T, Geuzaine C, Remacle J-F, Lambrechts J. Robust untangling of curvilinear meshes. J Comput Phys 2013;254:8–26.
[12] Gargallo-Peiró A, Roca X, Sarrate J. A surface mesh smoothing and untangling method independent of the CAD parameterization. Comput Mech 2014;53(4):587–609.
[13] Gargallo-Peiró A, Roca X, Peraire J, Sarrate J. Defining quality measures for validation and generation of high-order tetrahedral meshes. In: Proceedings of the 22nd international meshing roundtable. Springer International Publishing; 2014. p. 109–26.
[14] Roca X, Gargallo-Peiró A, Sarrate J. Defining quality measures for high-order planar triangles and curved mesh generation. In: Proceedings of the 20th international meshing roundtable. Springer Berlin Heidelberg; 2011. p. 365–83.
[15] Garanzha V. Variational principles in grid generation and geometric modelling: theoretical justifications and open problems. Numer Linear Algebra Appl 2004;11(5–6):535–63.
[16] Huang W, Russell R. Adaptive moving mesh methods. Springer; 2011.
[17] Sastry SP, Zala V, Kirby RM. Thin-plate-spline curvilinear meshing on a calculus-of-variations framework. Procedia Eng 2015;124:135–47.
[18] Garanzha V, Kudryavtseva L, Utyuzhnikov S. Variational method for untangling and optimization of spatial meshes. J Comput Appl Math 2014;269:24–41.
[19] Felippa CA. Optimization of finite element grids by direct energy search. Appl Math Model 1976;1(2):93–6.
[20] Turner M, Peiró J, Moxey D. A variational framework for high-order mesh generation. Procedia Eng 2016;82:127–35.
[21] Turner M, Moxey D, Sherwin SJ, Peiró J. Automatic generation of 3D unstructured high-order curvilinear meshes. In: Proceedings of the european congress on computational methods in applied sciences and engineering, Crete Island, Greece. 2016.
[22] Ball J. Convexity conditions and existence theorems in non-linear elasticity. Arch Ration Mech Anal 1977;63(4):337–403.
[23] Balzani D, Neff P, Schröder J, Holzapfel G. A polyconvex framework for soft biological tissues. Adjustment to experimental data. Int J Solids Struct 2006;43:6052–70.
[24] Bonet J, Gil A, Ortigosa R. A computational framework for polyconvex large strain elasticity. Comput Methods Appl Mech Engrg 2015;283:1061–94.
[25] Burger M, Modersitzki J, Ruthotto L. A hyperelastic regularization energy for image registration. SIAM J Sci Comput 2013;35(1):B132–48.
[26] Moxey D, Ekelschot D, Keskin Ü, Sherwin S, Peiró J. High-order curvilinear meshing using a thermo-elastic analogy. Comput Aided Des 2016;72:130–9.
[27] Holzapfel G. Nonlinear solid mechanics. Wiley; 2000.
[28] Bonet J, Wood R. Nonlinear continuum mechanics for finite element analysis. Cambridge University Press; 1997.
[29] Charakhch'yan A, Ivanenko S. A variational form of the Winslow grid generator. J Comput Phys 1997;136(2):385–98.
[30] Escobar J, Rodríguez E, Montenegro R, Montero G, González-Yuste J. Simultaneous untangling and smoothing of tetrahedral meshes. Comput Methods Appl Mech Engrg 2003;192(25):2775–87.
[31] Garanzha V, Kaporin I. Regularization of the barrier variational method of grid generation. Comput Math Math Phys 1999;39(9):1489–503.

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.

[32] Cantwell CD, Moxey D, Comerford A, Bolis A, Rocco G, Mengaldo G, et al. Nektar++: An open-source spectral/hp element framework. Comput Phys Comm 2015;192:205–19. http://dx.doi.org/10.1016/j.cpc.2015.02.008.
[33] Hesthaven JS, Warburton T. Nodal discontinuous Galerkin methods. Springer; 2008.
[34] Karniadakis GE, Sherwin S. Spectral/hp element methods for computational fluid dynamics. 2nd ed. OUP; 2005.
[35] Nocedal J, Wright S. Numerical optimization. 2nd ed. Springer series in operations research, Springer; 2006.
[36] Moxey D, Cantwell C, Kirby R, Sherwin S. Optimising the performance of the spectral/hp element method with collective linear algebra operations. Comput Methods Appl Mech Engrg 2016;310:628–45.
[37] Heinecke A, Pabst H, Henry G. LIBXSMM: A high performance library for small matrix multiplications. In: Supercomputing 2015, Austin, Texas. 2015.
[38] Witherden F, Vincent P. On the identification of symmetric quadrature rules for finite element methods. Comput Math Appl 2015;69(10):1232–41.
[39] Wandzurat S, Xiao H. Symmetric quadrature rules on a triangle. Comput Math Appl 2003;45(12):1829–40.
[40] Ruiz-Gironés E, Roca X, Sarrate J. High-order mesh curving by distortion minimization with boundary nodes free to slide on a 3D CAD representation. Comput Aided Des 2016;72:52–64.
[41] Moxey D, Green MD, Sherwin SJ, Peiró J. An isoparametric approach to high-order curvilinear boundary-layer meshing. Comput Methods Appl Mech Engrg 2015;283:636–50.
[42] Schneider R, Jimack PK. On the evaluation of finite element sensitivities to nodal coordinates. Electron Trans Numer Anal 2008;32:134–44.

Please cite this article in press as: Turner M., et al. Curvilinear mesh generation using a variational framework. Computer-Aided Design (2017), https://doi.org/10.1016/j.cad.2017.10.004.