J Eng Math (2006) 56:247–262
DOI 10.1007/s10665-006-9086-6

ORIGINAL PAPER

An explicit construction of interpolation nodes on the simplex

T. Warburton

Received: 28 September 2005 / Accepted: 20 July 2006 /
Published online: 27 September 2006
© Springer Science+Business Media B.V. 2006

Abstract An open question concerns the spatial distribution of nodes that are suitable for high-order Lagrange interpolation on the triangle and tetrahedron. Several current methods used to produce nodal sets with small Lebesgue constants are recalled. A new approach is presented for building nodal distributions of arbitrary order, that is based on curvilinear finite-element techniques. Numerical results are shown which demonstrate that, despite the explicit nature of this construction, the resulting node sets are well suited for interpolation and competitive with existing sets for up to tenth-order polynomial interpolation. Matlab scripts which evaluate the node distributions on the equilateral triangle are included.

Keywords Simplex · Interpolation · Nodes · Fekete · Lebesgue constant

1 Introduction

An open question concerns the spatial distribution of nodes that are suitable for high-order Lagrange interpolation on the triangle and tetrahedron. There have been several attempts to produce such nodal sets by direct and indirect methods that minimize the Lebesgue constant. We will summarize existing nodal sets and then discuss a simple approach that yields nodes possessing surprisingly good Lebesgue constants for up to tenth-order polynomial interpolation.

An early approach introduced in (1), involves choosing node locations that maximize the determinant of the classical Vandermonde matrix defined in terms of a bivariate monomial basis up to seventh order. The resulting nodes are referred to as Fekete nodes (2). This approach was extended up to thirteenth order in (3), and further extended to nineteenth order in (4). The latter group used a steepest-descent algorithm and explicit formulae for computing the gradient of the determinant of a generalized Vandermonde matrix. Recently, (5) and (6) independently presented more sophisticated optimization techniques and revisited the algorithms used to find nodal sets, starting with Fekete distributions and generating configurations with improved Lebesgue constants.

An alternative, physically motivated, approach was taken in (7) based on the observation of Stieltjes (8,9) that the location of the roots of certain Jacobi polynomials coincide with the equilibrium configuration

T. Warburton (✉)
Computational and Applied Mathematics, Rice University, 6100 Main Street MS-134, Houston,
TX 77005-1892, USA
e-mail: timwar@rice.edu

of charges constrained to lie on the bi-unit interval. This observation immediately implies that the Gauss–Lobatto–Legendre quadrature points coincide with equilibrium positions of these charges. Hesthaven (7) extended this analogy to compute node distributions in the triangle by seeking equilibrium positions of charges distributed in the triangle with line charges set on the boundary of the triangle.

Thus far, we have discussed node distributions resulting from optimizing the interpolation quality of nodes by varying their location. By contrast (10) introduced node distributions that have explicit formulae for their construction. This is of great practical interest, as well as reveals basic structural properties of the distributions that are responsible for the interpolation quality demonstrated by the optimization-based node sets discussed above. A similar philosophy and approach was used in (11).

In this paper, we introduce a simple construction that allows us to produce node distributions for arbitrary-order interpolation. The resulting distributions are comparable with all of the above distributions up to tenth order. We will describe a formula that has one tunable parameter and completely describes some possible nodal distributions of any given order. The basic philosophy is to replace the task of creating a nodal distribution with the closely related task of building a coordinate-warping transformation for the triangle. This is a familiar problem faced when using curvilinear finite elements (12). New contributions of this work include the application of the *warp & blend* transform to create nodal elements, and the construction of coordinate transforms that do not actually change the overall geometry of the triangle.

We settled on a one-parameter family of nodes, for simplicity and because our experiments revealed surprisingly good interpolation properties when the parameter is tuned in a straightforward manner. We shall describe a simple generalization to the tetrahedron, and finally present experimental results demonstrating the effectiveness of the node distributions.

2 Formulation

For convenience, we consider an equilateral reference triangle with edges of length two centered at the origin. In Fig. 1, we show the location of the vertex nodes and define the so-called barycentric coordinates. These are a triad of coordinates for a point (x, y) related to the areas of the triangles created by partitioning the triangle into three subtriangles, each containing two edge vertices and the point itself as a third vertex.

We can describe any point (x, y) in the equilateral triangle as a linear combination of the barycentric coordinates,

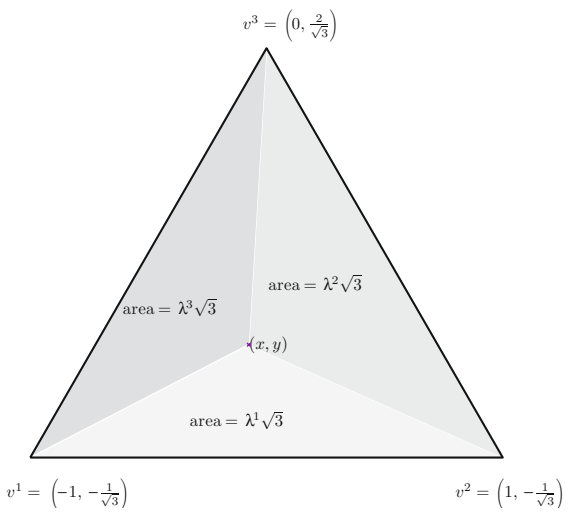

Fig. 1 Triangle vertex coordinates for an equilateral triangle, and definition of the barycentric coordinates for a point (x, y) in the triangle

$$(x, y) = \lambda^2 \mathbf{v}^1 + \lambda^3 \mathbf{v}^2 + \lambda^1 \mathbf{v}^3, \tag{1}$$

with the restriction that $\lambda^1 + \lambda^2 + \lambda^3 = 1$.

In order to perform polynomial interpolation, we require a finite polynomial space. For this study, we chose the space of polynomials of total degree at most p on the triangle T, denoted by

$$\mathbf{P}^p(T) = \left\langle x^i y^j | 0 \le i + j \le p \right\rangle.$$

The dimension of this space is $N_p = (p + 1)(p + 2)/2$. A convenient, orthogonal and stably computable basis for this space was proposed in (13). The basis functions are:

$$\phi_{n_{ij}}\left(\lambda^1, \lambda^2, \lambda^3\right) = P_i^{0,0}\left(\frac{\lambda^3 - \lambda^2}{\lambda^3 + \lambda^2}\right)\left(\lambda^3 + \lambda^2\right)^i P_j^{2i+1,0}\left(2\lambda^1 - 1\right), \tag{2}$$

where

$$n_{ij} = 1 + \left\{\frac{(i + j)\,(i + j + 1)}{2}\right\} + j, \quad 0 \le i, j; i + j \le p. \tag{3}$$

Given N_p interpolation points on the triangle $\left\{\lambda_n^1, \lambda_n^2, \lambda_n^3 \,|1 \le n \le N_p\right\}$, we can construct a generalized Vandermonde matrix

$$\mathbf{V}_{nm} = \phi_m\left(\lambda_n^1, \lambda_n^2, \lambda_n^3\right), \tag{4}$$

which provides us with a compact way of representing the pth-order, polynomial, Lagrange interpolating functions $\left\{l_n \,|1 \le n \le N_p\right\}$ associated with the N_p interpolation points. Assuming the generalized Vandermonde matrix is invertible, we can use the following result

$$l_n\left(\lambda^1, \lambda^2, \lambda^3\right) = \sum_{m=1}^{m=N_p} \left(\mathbf{V}^{-1}\right)_{mn} \phi_m\left(\lambda^1, \lambda^2, \lambda^3\right). \tag{5}$$

This definition is not required to generate the *warp & blend* nodal sets . However, it is useful in computing their Lebesgue constant, as described in the next section.

2.1 Lebesgue constant

We first introduce the Lebesgue function

$$\lambda\left(\lambda^1, \lambda^2, \lambda^3\right) = \sum_{n=1}^{n=N_p} \left|l_n\left(\lambda^1, \lambda^2, \lambda^3\right)\right|, \tag{6}$$

which is unitary at the interpolation nodes and define the Lebesgue constant, Λ_p, on a triangle T

$$\Lambda_p = \max_{(\lambda^1, \lambda^2, \lambda^3) \in T} \lambda\left(\lambda^1, \lambda^2, \lambda^3\right).$$

The Lebesgue constant is considered an interpolation quality indicator for a set of nodes because of its appearance in the following error estimate

$$\|u - I_p u\|_\infty \le (1 + \Lambda_p) \inf_{\tilde{u} \in \mathbf{P}^p(T)} \|u - \tilde{u}\|_\infty,$$

where I_p is the pth-order interpolation operator based on the node set in question. This result is easily derived by observing that, in fact, the Lebesgue constant is also the operator max-norm of I_p. This result puts an upper bound on the worst possible pointwise error for the polynomial interpolant relative to the best possible pointwise polynomial approximation available at the same order. It is thus reasonable to construct a set of nodes whose Lebesgue constant is as small as possible.

When reporting Lebesgue constants later in this paper, we actually report approximate values. We have applied the following directed random sampling approach in an attempt to find the global maximizer of the Lebesgue function:

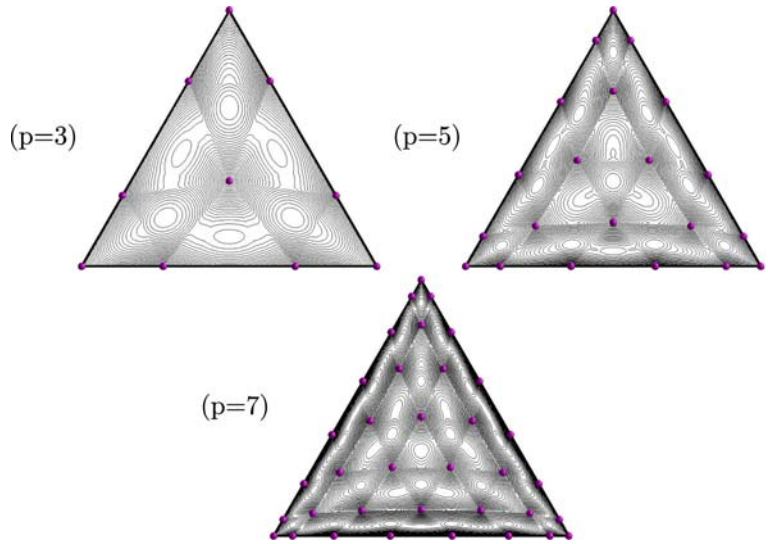

Fig. 2 Isocontour plots of the Lebesgue functions for three different sets of nodes (α optimal *warp* & *blend* nodes computed in Sect. 3.1)

1. Make a *list of current sample points* of $N = 10{,}000$ randomly chosen in the triangle.
2. For $i = 1$ to 20:
   (a) Evaluate the Lebesgue function at the *list of current sample points*.
   (b) Find the 10% of sample points with the largest Lebesgue function value.
   (c) For each of the top 10% sample create 10 new random sample locations within a box of side $(1/2)^{i-2}$ centered at the sample.
   (d) Replace the *list of current sample points* with a combination of the top 100 samples points and the newly created sample points that happen to be located inside the triangle.
3. Evaluate the Lebesgue function at the *list of current sample points*.
4. Report the largest value as the approximate Lebesgue constant.

This sequence requires approximately 200,000 evaluations of the Lebesgue function, which is considerable; fortunately we do not need to evaluate this number frequently, even in the optimization process to be described later. A modest workstation completes the computation in short order. In experiments, this algorithm reported reliable estimates of the Lebesgue constant. A more efficient algorithm was reported recently in (6).

The piecewise smoothness and limited number of maxima of the Lebesgue function contribute to the success of the algorithm. In Fig. 2, we show isocontours of the Lebesgue function for sets of nodes at different orders. As expected the nodes are located at minima of the Lebesgue function. We also note that there are visible creases in the function, which lie on curved lines joining the original equi-spaced nodes. Finally, in general, there appears to be a limited number of maxima associated with curvilinear triangles joining nearby nodes.

2.2 Nodal distribution construction

Given any set of N_p node coordinates $\{\mathbf{x}^i \mid 1 \le i \le N_p\}$, it is always possible to construct a 2-vector function $\mathbf{f}$ defined on the triangle with the property that:

$$\mathbf{f}\left(\mathbf{x}^{e,i}\right) = \mathbf{x}^i \text{ for all } 1 \le i \le N_p \tag{7}$$

where $\{\mathbf{x}^{e,i} \mid 1 \leq i \leq N_p\}$ is the set of equi-spaced nodes. Furthermore, there is a unique 2-vector function in $\mathbf{P}^p(T) \times \mathbf{P}^p(T)$ that satisfies Eq. 7.

This simple observation frames the question of constructing a set of nodes in terms of such a transform. Our first experiments in constructing an isoparametric polynomial transform yielded nodes that possess surprisingly good Lebesgue constants. Subsequently, we relaxed the condition that the transform be of the same order as the original equi-spaced nodal set, and we were able to compute nodal sets with improved Lebesgue constants.

To narrow down the choice of how to construct the isoparametric transform, we require the following properties:

1. The image of the equi-spaced nodes on an edge should be a Gauss–Lobatto–Legendre distribution.
2. The transform should be bi-jective.
3. The transform should be symmetric with respect to the symmetries of the origin equilateral triangle.
4. The transform should be explicit in the barycentric coordinates.

We can view the task of creating such a transform as an exercise in creating an isoparametric curvilinear finite element (12). However, in this context, we are creating a curvilinear element by deforming the reference triangle without changing its outline geometry. For clarity, we consider the deformation required of the lower edge of the equilateral triangle and repeat a similar deformation for the other two edges.

Given the locations of the $p+1$ point Gauss–Lobatto–Legendre quadrature nodes on the biunit interval: $\{x^i \mid 1 \leq i \leq p+1\}$, we construct a one-dimensional deformation function $w : [-1,1] \to [-1,1]$ such that

$$w(x) = \sum_{i=1}^{i=p+1} \left(x^i - x^{e,i}\right) \prod_{j=1, j\neq i}^{j=p+1} \left(\frac{x - x^{e,j}}{x^{e,i} - x^{e,j}}\right),$$

where $x^{e,i} = (-p + 2i - 2)/p$. This is nothing more than the Lagrange interpolant function which interpolates the deformation required for mapping the equi-spaced points to the the GLL points.

In Fig. 3, we plot $x + w(x)$, i.e., the effect of the warp function. It is interesting to note that the coordinate warping for $p = 3, 5, 20$ are all very similar.

We extend the edge warp into the triangle by blending in the edge normal direction. In preparation for satisfying the symmetry requirements on the transform, we express the deformation transform for the blended lower edge in terms of the barycentric coordinates:

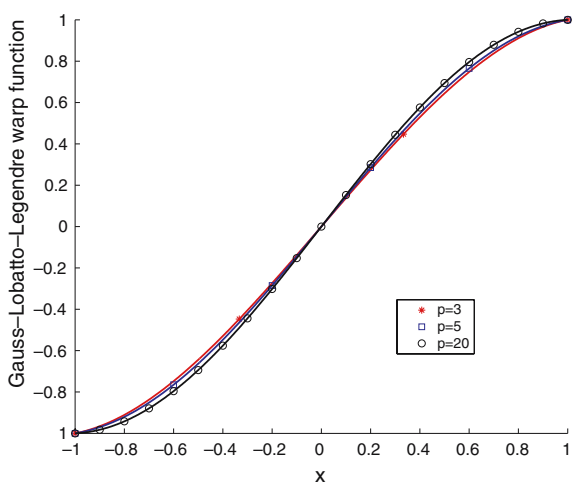

Fig. 3 Warp functions $x + w(x)$ used for $p = 3, 5, 20$ designed to interpolate the Gauss–Lobatto–Legendre node locations at equally spaced nodes in $[-1, 1]$

$$\mathbf{w}^1\left(\lambda^1,\lambda^2,\lambda^3\right) = w\left(\lambda^3-\lambda^2\right),(1,0) \tag{8}$$

$$b^1\left(\lambda^1,\lambda^2,\lambda^3\right) = \left(\frac{2\lambda^3}{2\lambda^3+\lambda^1}\right)\left(\frac{2\lambda^2}{2\lambda^2+\lambda^1}\right). \tag{9}$$

Multiplying these two together, we achieve a *warp & blend* transform $\mathbf{g}^1 = b^1\mathbf{w}^1$ that satisfies our coordinate transform requirements for edge one. The apparent singularities in the blending function at $\lambda^1 = \lambda^3 = 0$ and also $\lambda^1 = \lambda^2 = 0$ are, in fact, removed in the product because both locations coincide with roots of $\mathbf{w}^1$. Thus we trivially observe that this *warp & blend* function is a 2-vector polynomial of degree at most p.

The warping functions for the other two edges can be expressed by permuting $\lambda^1, \lambda^2, \lambda^3$

$$\mathbf{w}^2\left(\lambda^1,\lambda^2,\lambda^3\right) = w\left(\lambda^1-\lambda^3\right)\left(-\frac{1}{2},\sqrt{\frac{3}{2}}\right), \tag{10}$$

$$\mathbf{w}^3\left(\lambda^1,\lambda^2,\lambda^3\right) = w\left(\lambda^2-\lambda^1\right)\left(-\frac{1}{2},-\sqrt{\frac{3}{2}}\right). \tag{11}$$

The associated blending functions are

$$b^2\left(\lambda^1,\lambda^2,\lambda^3\right) = \left(\frac{2\lambda^3}{2\lambda^3+\lambda^2}\right)\left(\frac{2\lambda^1}{2\lambda^1+\lambda^2}\right), \tag{12}$$

$$b^3\left(\lambda^1,\lambda^2,\lambda^3\right) = \left(\frac{2\lambda^2}{2\lambda^2+\lambda^3}\right)\left(\frac{2\lambda^1}{2\lambda^1+\lambda^3}\right). \tag{13}$$

Finally, we can express the coordinate shift as

$$\mathbf{g}\left(\lambda^1,\lambda^2,\lambda^3\right) = b^1\mathbf{w}^1 + b^2\mathbf{w}^2 + b^3\mathbf{w}^3. \tag{14}$$

In Sect. 3.1, we show results for this construction. However, given the flexibility of the construction of such a transform, we relaxed the constraint that the transform should be isoparametric. Examining the node descriptions, we observe that the nodes did not cluster particularly strongly around the triangle edges. This is because the blending functions drop to zero at the edges of the elements. A simple modification to the blending yields node sets with potentially better Lebesgue constants

$$\mathbf{g}\left(\lambda^1,\lambda^2,\lambda^3\right) = \left(1+\left(\alpha\lambda^1\right)^2\right)b^1\mathbf{w}^1 + \left(1+\left(\alpha\lambda^2\right)^2\right)b^2\mathbf{w}^2 + \left(1+\left(\alpha\lambda^3\right)^2\right)b^3\mathbf{w}^3.$$

In Fig. 4, we show plots of the vector functions $\mathbf{g}^1, \mathbf{g}^2, \mathbf{g}^3$ sampled at equi-spaced points in the triangle. In Sect. 6, we list three Matlab functions: `gll.m, warpfactor.m, nodes.m` that compute the Gauss–Lobatto–Legendre nodes, the warping factor with end-roots deflated, and the *warp & blend* nodes for the triangle, respectively.

2.3 Tetrahedron

The construction of the coordinate transform can be generalized for the equilateral tetrahedron. For each face of the tetrahedron, we use the transform constructed for the triangle. These transforms are then blended into the interior of the tetrahedron. The vector warp functions for each of the four faces are:

$$\mathbf{w}^1 = \mathbf{g}_1\left(\lambda^2,\lambda^3,\lambda^4\right)\mathbf{t}^{1,1} + \mathbf{g}_2\left(\lambda^2,\lambda^3,\lambda^4\right)\mathbf{t}^{1,2}, \tag{15}$$

$$\mathbf{w}^2 = \mathbf{g}_1\left(\lambda^1,\lambda^3,\lambda^4\right)\mathbf{t}^{2,1} + \mathbf{g}_2\left(\lambda^1,\lambda^3,\lambda^4\right)\mathbf{t}^{2,2}, \tag{16}$$

$$\mathbf{w}^3 = \mathbf{g}_1\left(\lambda^1,\lambda^4,\lambda^2\right)\mathbf{t}^{3,1} + \mathbf{g}_2\left(\lambda^1,\lambda^4,\lambda^2\right)\mathbf{t}^{3,2}, \tag{17}$$

$$\mathbf{w}^4 = \mathbf{g}_1\left(\lambda^1,\lambda^3,\lambda^2\right)\mathbf{t}^{4,1} + \mathbf{g}_2\left(\lambda^1,\lambda^3,\lambda^2\right)\mathbf{t}^{4,2}, \tag{18}$$

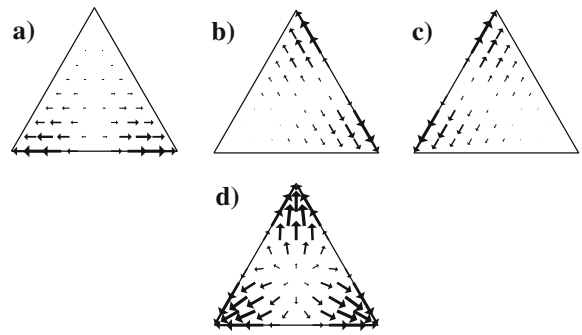

Fig. 4 (**a–c**): *Warp & blend* functions for each of the three edges, constructed with warping that reproduces a ten point Gauss–Lobatto–Legendre quadrature. (**d**) Sum of *warp & blend* functions for the three edges

where $(\lambda^1, \lambda^2, \lambda^3, \lambda^4)$ are the barycentric coordinates for the tetrahedron, the subscripts on the *warp & blend* $\mathbf{g}$ functions refer to the components of these functions and $\mathbf{t}^{f,1}, \mathbf{t}^{f,2}$ are two vectors forming orthogonal axes in the plane of face f specified for the origin centered equilateral tetrahedron by

$$\begin{aligned} &\mathbf{t}^{1,1} = (1,0,0)\,, \quad \mathbf{t}^{1,2} = (0,1,0)\,, \quad \mathbf{t}^{2,1} = (1,0,0)\,, \quad \mathbf{t}^{2,2} = \left(0, \tfrac{1}{3}, \sqrt{\tfrac{8}{9}}\right), \\ &\mathbf{t}^{3,1} = \left(-\tfrac{1}{2}, \sqrt{\tfrac{3}{4}}, 0\right), \quad \mathbf{t}^{3,2} = \left(-\sqrt{\tfrac{1}{12}}, \tfrac{1}{6}, \sqrt{\tfrac{8}{9}}\right), \quad \mathbf{t}^{4,1} = \left(\tfrac{1}{2}, \sqrt{\tfrac{3}{4}}, 0\right), \quad \mathbf{t}^{4,2} = \left(\sqrt{\tfrac{1}{12}}, -\tfrac{1}{6}, \sqrt{\tfrac{8}{9}}\right). \end{aligned} \tag{19}$$

The four, scalar, face, blending functions are given by

$$b^1 = \left(\frac{2\lambda^2}{2\lambda^2 + \lambda^1}\right)\left(\frac{2\lambda^3}{2\lambda^3 + \lambda^1}\right)\left(\frac{2\lambda^4}{2\lambda^4 + \lambda^1}\right), \tag{20}$$

$$b^2 = \left(\frac{2\lambda^1}{2\lambda^1 + \lambda^2}\right)\left(\frac{2\lambda^3}{2\lambda^3 + \lambda^2}\right)\left(\frac{2\lambda^4}{2\lambda^4 + \lambda^2}\right), \tag{21}$$

$$b^3 = \left(\frac{2\lambda^1}{2\lambda^1 + \lambda^3}\right)\left(\frac{2\lambda^2}{2\lambda^2 + \lambda^3}\right)\left(\frac{2\lambda^4}{2\lambda^4 + \lambda^3}\right), \tag{22}$$

$$b^4 = \left(\frac{2\lambda^1}{2\lambda^1 + \lambda^4}\right)\left(\frac{2\lambda^2}{2\lambda^2 + \lambda^4}\right)\left(\frac{2\lambda^3}{2\lambda^3 + \lambda^4}\right). \tag{23}$$

The combined *warp & blend* deformation is then

$$\mathbf{g}\left(\lambda^1, \lambda^2, \lambda^3, \lambda^4\right) = b^1\mathbf{w}^1 + b^2\mathbf{w}^2 + b^3\mathbf{w}^3 + b^4\mathbf{w}^4. \tag{24}$$

As with the triangle, we can increase the clustering of nodes near the boundary faces by modifying the blending functions

$$\mathbf{g}\left(\lambda^1, \lambda^2, \lambda^3, \lambda^4\right) = \left(1 + \left(\beta\lambda^1\right)^2\right) b^1\mathbf{w}^1 + \left(1 + \left(\beta\lambda^2\right)^2\right) b^2\mathbf{w}^2 + \left(1 + \left(\beta\lambda^3\right)^2\right) b^3\mathbf{w}^3 + \left(1 + \left(\beta\lambda^4\right)^2\right) b^4\mathbf{w}^4$$

for some parameter β. To retain a one-parameter set of nodes, that are easily optimized, we arbitrarily choose $\beta = \alpha$. There may be some benefit in allowing α and β to differ, and these options are being evaluated in ongoing studies.

3 Results

3.1 Triangle nodes

For the following experiment, we constructed the *warp & blend* nodes in two ways. First, we built the nodes with the blending parameter α set to zero. Next, we used the Matlab function `fminbnd`, which is

an optimization algorithm based on golden-section search and parabolic interpolation [14, 15] to choose an α which at least locally minimizes the Lebesgue constant computed with the algorithm described in Sect. 2.1, but with twenty-five-hundred initial samples and 10 iterations. This approach is reasonable, given the uncomplicated dependence of the Lebesgue constant on α, as demonstrated in Fig. 5 where we show the Lebesgue constant plotted as a function of α for $p = 8$ nodes.

After an optimal α was found, we recomputed the Lebesgue constant with 10,000 initial samples and 20 iterations. The values of α reported by `fminbnd` for a range of polynomial orders are reported in Table 7.

In Table 1, we compare the Lebesgue constants for these two sets of nodes with the nodes in (10) which are similarly given by explicit formulae. We see that the unoptimized nodes have comparable Lebesgue constants to the Blyth and Pozrikidis nodes (10). The optimized nodes show significant improvement. Furthermore, in Table 2, we report condition numbers for the generalized Vandermonde matrices (see Eq. 4) computed both with the Blyth and Pozrikidis nodes and with the *warp & blend* nodes. The condition number of the generalized Vandermonde matrix is important because of the pivotal role that the inverse of this matrix plays in constructing a Lagrange interpolating basis for the nodes as shown in Eq. 4. The condition numbers for both node types are comparable and indicate that we should expect to be able to compute the Lagrange interpolating functions with up to 13 digits of accuracy in double precision based arithmetic.

The *warp & blend* nodes are also comparable in interpolation quality to other classes of symmetric nodes that have Gauss–Lobatto–Legendre edge node distributions. In Table 3, we show comparisons with Roth's improved version of the Chen–Babuska nodes as well as his improved Fekete nodes (6). In addition, we include results from Heinrichs's approximate Lebesgue constant optimized nodes (5), and Hesthaven's electrostatics nodes (7).

In Fig. 6, we show optimized *warp & blend* node distributions for the triangle for a range of interpolation orders . Up to tenth order the distributions are very similar to the Fekete, electrostatic, Lebesgue-optimized, and other node sets currently known. It is also clear that, despite the simplicity of the node formula, it captures the basic behavior of all these other more complicated approaches.

Given the competitive Lebesgue constants of the new node distributions, it is also apparent that, although seemingly ad hoc, the new approach is as successful (up to tenth order) as most existing methods that instead involve computing node locations by optimizing proxies for the Lebesgue constant. One exceptional approach used in (6) is designed to optimize node locations by minimizing an efficiently and reliably computed approximation for the Lebesgue constant. One might view the new approach in this

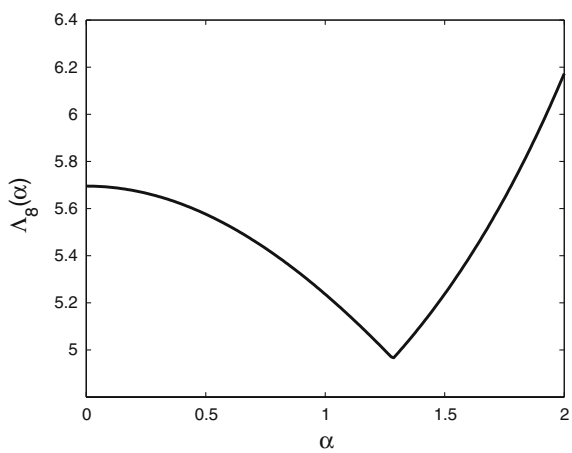

Fig. 5 The Lebesgue constant as a function of α for $p = 8$

Table 1 Lebesgue constants for Pozrikidis nodes compared with *warp & blend* nodes (computed with the algorithm given in Sect. 2.1)

Node class	Equi-spaced	Blyth and Pozrikidis (10)	*Warp & blend* $\alpha = 0$	*Warp & blend* α optimized
Λ_3	2.27	2.11	2.11	2.11
Λ_4	3.47	2.66	2.66	2.66
Λ_5	5.45	3.14	3.12	3.12
Λ_6	8.75	3.87	3.82	3.70
Λ_7	14.35	4.66	4.55	4.27
Λ_8	24.01	5.93	5.69	4.96
Λ_9	40.92	7.39	7.02	5.74
Λ_{10}	70.89	9.83	9.16	6.67
Λ_{11}	124.53	12.92	11.83	7.90
Λ_{12}	221.41	17.78	16.06	9.36
Λ_{13}	397.70	24.53	21.71	11.47
Λ_{14}	720.70	34.62	30.33	13.97
Λ_{15}	1315.9	49.46	42.48	17.65

Table 2 Condition numbers for generalized Vandermonde matrices formed using the orthonormalized basis (13)

Node class	Blyth and Pozrikidis (10)	*Warp & blend* α optimized
κ_3	5.9028	5.9028
κ_4	6.7763	6.7769
κ_5	7.7280	7.8450
κ_6	9.8423	9.5913
κ_7	11.4944	11.1597
κ_8	14.2101	13.8858
κ_9	18.0994	16.8957
κ_{10}	23.6271	21.6675
κ_{11}	31.4576	27.4011
κ_{12}	43.3978	36.1156
κ_{13}	61.0569	47.1973
κ_{14}	88.7706	63.6592
κ_{15}	130.2558	85.6918

work as a hybrid method that directly optimizes the Lebesgue constant but restricts the feasible node distributions to those given by a specified explicit formula, which has one free parameter.

In Table 3, we reported Lebesgue constants for existing sets of nodes that have Gauss–Lobatto–Legendre distributions and are symmetric. Roth (6) and Heinrichs (5), both created nodal sets with improved Lebesgue constants by dropping these conditions. We achieved minor improvements in the Lebesgue constant for the *warp & blend* nodes by changing the edge warp (i.e., changing the required nodal distributions on the edge) to an extended Chebychev distribution.

The *warp & blend* node sets are straightforward to compute at any order. However, as shown in Table 1, the Lebesgue constant for these nodes grows relatively quickly above $p = 10$. As has been reported for Fekete nodes and Roth's Lebesgue-constant optimized nodes, there is a significant change in the types of distribution results after optimization. The structure of their optimized interior node distributions becomes visibly less regular as the polynomial order is increased. This immediately raises the possibility that our explict formulae may benefit from additional terms and perhaps more parameters to capture the extra dynamic in the coordinate transform.

Table 3 Comparison of Lebesgue constants for a range of current symmetric nodal distributions on the triangle with Gauss–Lobatto–Legendre distributions on edges

Node class	*Warp & blend* [c] α optimized	Chen–Babuska[a]	Fekete[a]	Heinrichs[b]	Hesthaven[c]
Λ_3	2.11	2.111	2.113	–	2.113
Λ_4	2.66	2.692	2.729	–	2.588
Λ_5	3.12	3.301	3.611	–	3.196
Λ_6	3.70	3.791	4.171	3.87	4.075
Λ_7	4.27	4.391	4.928	–	4.780
Λ_8	4.96	5.089	5.905	–	5.855
Λ_9	5.74	5.918	6.803	5.89	6.886
Λ_{10}	6.67	7.085	7.852	–	8.466
Λ_{11}	7.90	7.266	7.907	–	10.144
Λ_{12}	9.36	8.669	8.472	7.59	12.637
Λ_{13}	11.47	9.291	9.279	–	–
Λ_{14}	13.97	8.988	9.959	–	–
Λ_{15}	17.65	10.306	10.021	9.25	–

Lebesgue constants computed by: [a] (6), [b] (5), [c] by algorithm in Sect. 2.1

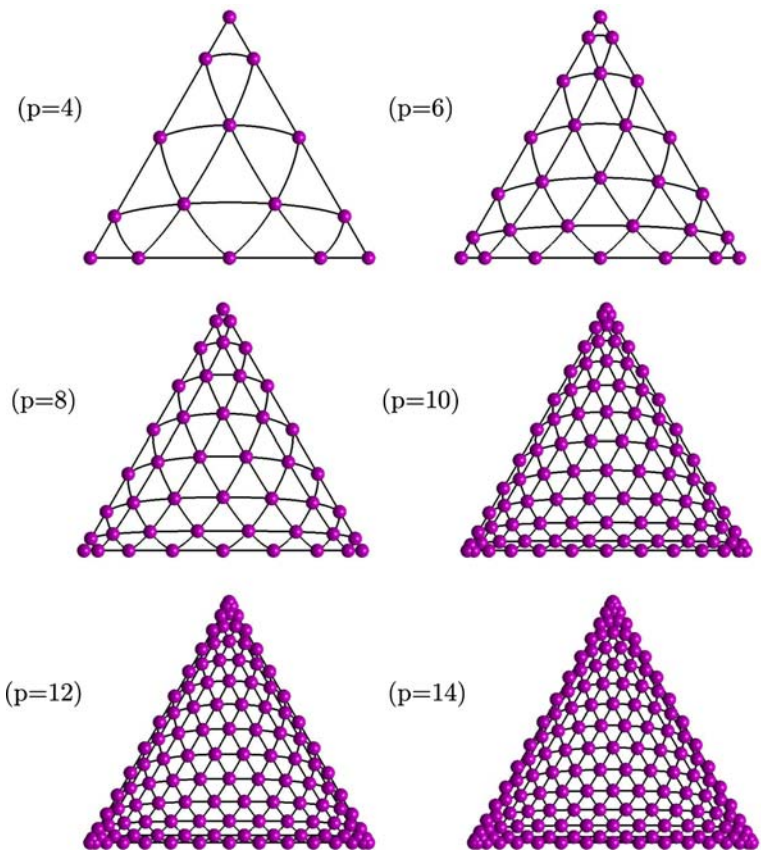

Fig. 6 Node distributions for $p = 4, 6, 8, 10, 12, 14$. The black lines represent the effect of the *warp & blend* transform on the lines of constant $\lambda^1, \lambda^2,$ or λ^3 that connect the undeformed equi-distributed nodes

3.2 Tetrahedron nodes

The number of different node distribution types available for the tetrahedron is markedly smaller than those known for the triangle. In Table 4, we show Lebesgue constants for equally spaced nodes, our un-optimized and optimized *warp & blend* nodes, Hesthaven and Teng's electrostatic nodes, and the nodes of Chen and Babuska. We are again surprised that the *warp & blend* nodes are so competitive with available node sets. In Fig. 7, we show α optimized *warp & blend* node sets for the tetrahedron for $p = 6, 8, 10, 12$.

4 Interpolation tests

We have thus far used the Lebesgue constant of nodal sets as an interpolation quality indicator. To further compare some of the nodal sets already discussed, we adopt two interpolation tests proposed in (5). In that paper the nodal sets were used to interpolate the following two functions on the bi-unit triangle

Table 4 Comparison of Lebesgue constants for a range of current nodal distributions on the tetrahedron

Node class	*warp & blend* α optimized	Hesthaven and Teng (16)	Chen and Babuska (17)	Equi-spaced
Λ_4	4.07	4.0774	4.1120	4.88
Λ_5	5.32	5.3470	5.6158	8.09
Λ_6	7.01	7.3391	7.3632	13.66
Λ_7	9.21	9.7588	9.3659	23.38
Λ_8	12.54	13.626	12.311	40.55
Λ_9	17.02	18.901	15.659	71.15
Λ_{10}	24.36	27.190	–	126.20
Λ_{11}	36.35	–	–	225.99
Λ_{12}	54.18	–	–	409.15
Λ_{13}	84.62	–	–	742.69
Λ_{14}	135.75	–	–	1360.49
Λ_{15}	217.70	–	–	2506.95

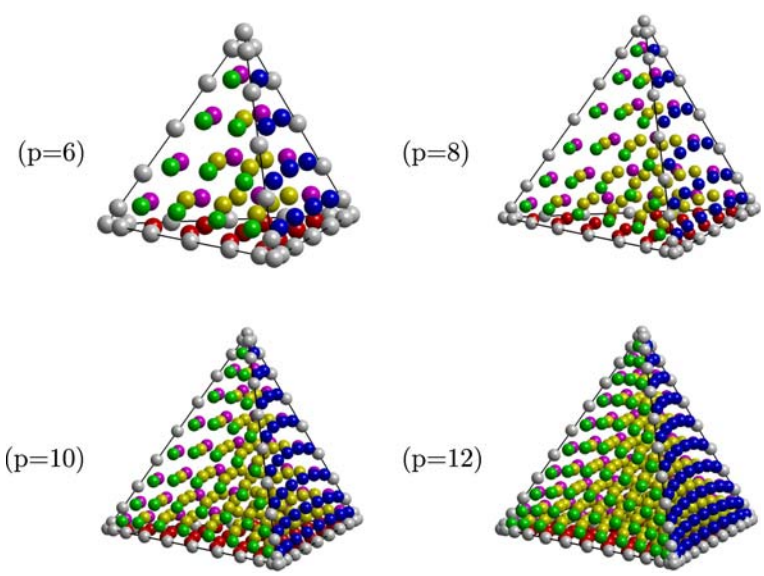

Fig. 7 α-optimized *warp & blend* tetrahedral node distribution for $p = 6, 8, 10, 12$

Table 5 Maximum interpolation errors for Test I computed with directed random search excluding (∗) results which are from (5)

p	Blyth and Pozrikidis (10)	*Warp & blend*	Heinrichs (5)	Hesthaven (7)
6	5.7760e−005	5.3819e−005	5.6005e−005	5.9476e−005
9	3.8020e−008	2.7592e−008	2.5150e−008	3.3607e−008
12	1.0403e−011	5.1230e−012	1.7273e−012	–
15	1.8596e−015	1.1380e−015	6.25e−015 (∗)	–
18	1.7087e−015	1.1519e−015	1.28e−014 (∗)	–

Table 6 Maximum interpolation errors for Test II computed with directed random search excluding (∗) results which are from (5)

p	Blyth and Pozrikidis (10)	*warp & blend*	Heinrichs (5)	Hesthaven (7)
6	6.9077e−004	6.4327e−004	6.5509e−004	7.2943e−004
9	1.0526e−006	7.6282e−007	6.8984e−007	9.6364e−007
12	2.2461e−010	1.1033e−010	4.4139e−011	–
15	4.9061e−014	1.8077e−014	3.09e−014 (*)	–
18	1.6029e−014	5.4401e−015	7.71e−014 (*)	

$T = \{-1 \leq x, y; x + y \leq 0\}$:

Test I: $u(x, y) = (x + 1)(y + 1)\left(e^{x+y} - 1\right)$,

Test II: $u(x, y) = (x + 1)(y + 1)(\cosh(x + y) - 1)$,

where we used a variation of the algorithm described in Sect. 2.1 to compute an approximation of the maximum pointwise error of the $p = 6, 9, 12, 15, 18$ polynomial interpolant for these two functions.

In Table 5, we compare the maximum pointwise errors for the *warp & blend* nodes compared with the nodal sets of (5,7,10) applied to Test I. We see that, in general, the ratio between errors is related to their relative Lebesgue constants. The non-symmetric $p = 12$ nodes of Heinrichs are noticeably better than the other $p = 12$ node sets, this we attribute to their smaller Lebesgue constant. We included results for $p = 15, 18$ to complete the comparison with Heinrichs's results. However, it is clear from the results and inspecting the solution error that the error is dominated by finite precision effects.

In Table 6, we show similar comparisons for Test II and note briefly that results are consistent with Test I. As before, the *warp & blend* nodes degrade in performance for $p = 12$, but this is consistent with the Lebesgue constant quality measure for these nodes being larger than Heinrichs's nodes (5).

5 Conclusions

In conclusion, we have presented a simple and geometrically motivated way of spatially distributing interpolation nodes in a triangle and tetrahedron. The method we used was inspired by techniques employed in the creation of curvilinear finite-element methods and represents a new approach and perspective on the problem of creating these nodal sets.

The new *warp & blend* nodes were shown to have Lebesgue constants which are better than or comparable with all existing node sets up to at least tenth-order interpolation. We showed test interpolation results that give further confidence on the quality of the *warp & blend* nodes for interpolation. The simplicity of the construction suggests we have captured the essence of the existing nodal distribtions in a simple way.

Finally, we have included short and simple Matlab scripts which generate the nodal sets, on a reference equilateral triangle, for arbitrary order and also reflect the simplicity of their construction.

Acknowledgements The work of the author was partially supported by the NSF under contract NSF-CNS-0324911, NSF under contract NSF-DMS-0512673, and AFOSR under contract FA9550-05-1-0473.

Appendix A

A.1 Gauss–Lobatto–Legendre nodes

To compute the location of the Gauss–Lobatto–Legendre nodes on the biunit interval, we follow Numerical Recipes (18). Included here is a Matlab script that uses Newton–Raphson iteration, to find the roots of $\left(1-x^2\right) P_p^{0,0}(x)$. Original Matlab implementation is provided by Greg von Winckel.

```
% (gll.m)
% input: polynomial order
% output: p+1 point Gauss-Lobatto-Legendre nodes.

function x=gll(p)

% polynomial order + 1
N=p+1;

% Use the Chebyshev-Gauss-Lobatto nodes as the first guess
x=cos(pi*(0:p)/p)';

% The Legendre Vandermonde Matrix
V=zeros(N,N);

xold=2;

% Newton-Raphson iteration
while max(abs(x-xold))>eps
 xold=x;

 V(:,1)=1;
 V(:,2)=x;
 for k=2:p
 V(:,k+1)=( (2*k-1)*x.*V(:,k)-(k-1)*V(:,k-1) )/k;
 end

 x=xold-( x.*V(:,N)-V(:,p) )./( N*V(:,N) );
end
```

A.2 Edge warp-factor function

We list a Matlab function that computes **g** as described in Sect. 2.2:

```
% (warpfactor.m)
% input: polynomial order
% input: vector of node locations to interpolate
% input: location of nodes to interpolate warp function at
% output: warp function at xout, with -1,+1 roots deflated
function warp = warpfactor(p, xnodes, xout)
  warp = zeros(size(xout));
  xeq = linspace(1,-1,p+1)';
```

```
  for i=1:p+1
    d = (xnodes(i)-xeq(i));
    for j=2:p
      if(i~=j)
    d = d.*(xout-xeq(j))/(xeq(i)-xeq(j));
      end
    end

  % deflate end roots
  if(i~=1)
    d = -d/(xeq(i)-xeq(1));
  end
  if(i~=(p+1))
    d = d/(xeq(i)-xeq(p+1));
  end

  warp = warp+d;
end
```

0.3 A *warp & blend* node construction

We list a Matlab function that computes the *warp & blend* nodes on an equilateral triangle. It depends on `gll.m` and `warpfactor.m` listed above.

```
% (nodes.m)
% input: p=polynomial order of interpolant
% input: alpha=tunable warping parameter
% output: X,Y vectors of node coordinates in equilateral triangle
function [X,Y] = nodes(p, alpha)

% total number of nodes
N = (p+1)*(p+2)/2;

% 1) compute Gauss-Lobatto-Legendre node distribution
gaussX = gll(p)

% 2) create equidistributed nodes on equilateral triangle
L1 = zeros(N,1); L2 = zeros(N,1); L3 = zeros(N,1);
sk = 1;
for n=1:p+1
 for m=1:p+2-n
 L1(sk) = (n-1)/p;
 L3(sk) = (m-1)/p;
 sk = sk+1;
 end
end
L2 = 1-L1-L3;
X = -L2 + L3;
Y = (-L2-L3+2*L1)/sqrt(3);

% 3) compute blending function at each node for each edge
blend1 = 4*L2.*L3;
blend2 = 4*L1.*L3;
blend3 = 4*L1.*L2;

% 4) amount of warp for each node, for each edge
warpfactor1 = warpfactor(p, gaussX, L3-L2);
warpfactor2 = warpfactor(p, gaussX, L1-L3);
```

```
warpfactor3 = warpfactor(p, gaussX, L2-L1);

% 5) combine blend & warp
warp1 = blend1.*warpfactor1.*(1 + (alpha*L1).^2);
warp2 = blend2.*warpfactor2.*(1 + (alpha*L2).^2);
warp3 = blend3.*warpfactor3.*(1 + (alpha*L3).^2);

% 6) accumulate deformations associated with each edge
X = X + 1*warp1 + cos(2*pi/3)*warp2 + cos(4*pi/3)*warp3;
Y = Y + 0*warp1 + sin(2*pi/3)*warp2 + sin(4*pi/3)*warp3;
```

Table 7 Locally optimal choices of the blending parameter α

Element	Triangle	Tetrahedron
α_3	1.4152	0.0000
α_4	0.1001	0.1002
α_5	0.2751	1.1332
α_6	0.9808	1.5608
α_7	1.0999	1.3413
α_8	1.2832	1.2577
α_9	1.3648	1.1603
α_{10}	1.4773	1.0153
α_{11}	1.4959	0.6080
α_{12}	1.5743	0.4523
α_{13}	1.5770	0.8856
α_{14}	1.6223	0.8717
α_{15}	1.6258	0.9655

0.4 Optimized choice of the α blending parameter

Table 7 indicates values, to four decimal places, for the α parameter that the Matlab function `fminbnd` found as possible minimizers for the Lebesgue constant computed as described in Sect. 2.1.

References

1. Bos LP (1983) Bounding the Lebesgue function for Lagrange interpolation in a simplex. J Approx Theory 38:43–59
2. Fekete M (1923) Über die verteilung der wurzeln bei gewissen algebraischen gleichungen mit ganzzahligen koeffizienten. Math Zeit 17:228–249
3. Chen Qi, Babuska Ivo (1995) Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle. Comp Meth App Mech Eng 128:405–417
4. Taylor M, Wingate B, Vincent RE (2000) An algorithm for computing fekete points in the triangle. SIAM J Num Anal 38:1707–1720
5. Heinrichs W (2005) Improved Lebesgue constants on the triangle. J Comp Phys 207:625–638
6. Roth MJ (2005) Nodal configurations and Voronoi Tessellations for triangular spectral elements. PhD thesis, University of Victoria
7. Hesthaven JS (1998) From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex. SIAM J Numer Anal 35:655–676
8. Stieltjes TJ (1885) Sur quelques théorèmes d'algèbre. Comptes Rendus de l'Academie des Sciences 100:439–440
9. Stieltjes TJ (1885) Sur les polynômes de jacobi. Comptes Rendus Acad Sci 100:620–622
10. Blyth MG, Pozrikidis C (2005) A Lobatto interpolation grid over the triangle. IMA J Appl Math 153–169
11. Bittencourt ML (2005) Fully tensorial nodal and modal shape functions for triangles and tetrahedra. Int J Numer Meth Eng 63:1530–1558

12. Gordon WN, Hall CA (1973) Construction of curvilinear coordinate systems and application to mesh generation. Int J Num Meth Eng 7:461–477
13. Proriol J (1957) Sur une Famille de Polynomes à deux Variables Orthogonaux dans un Triangle. C R Acad Sci Paris 257:2459–2461
14. Forsythe GE, Malcolm MA, Moler CB (1976) Computer methods for mathematical computations. Prentice-Hall
15. Brent RP (1973) Algorithms for minimization without derivatives. Prentice-Hall
16. Hesthaven JS, Teng CH (2000) Stable spectral methods on tetrahedral elements. SIAM J Sci Comp 21:2352–2380
17. Chen Qi, Babuska Ivo (1996) The optimal symmetrical points for polynomial interpolation of real functions in the tetrahedron. Comp Meth App Mech Eng 137:89–94
18. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) Numerical recipes in C: The art of computing scientific, 2nd edn. Cambridge University Press