Chapter 14

STABILIZED METHODS

14.1 Introduction

We have employed several weighted residual methods to formulate our finite element solutions. Most of the time we have used the standard Galerkin method where we multiply a residual error by a special weighting function. Recall that within an element we assumed a spatial interpolation for the approximate solution as

$$x \in \Omega^e : \qquad u(\boldsymbol{x}) = \boldsymbol{H}(\boldsymbol{x})\, \boldsymbol{U}^e \;=\; \sum_j H_j\,(\boldsymbol{x})\, U_j^e \tag{14.1}$$

which in turn defines the residual error

$$R\,(\boldsymbol{H}(x)) \neq 0 \tag{14.2}$$

The standard Galerkin method (sometimes called the *Bubnov-Galerkin)* is said to be a process that "makes the spatial approximation orthogonal to the residual error" by requiring the weighed system to be

$$\int_\Omega W_j\; R(\boldsymbol{x})\; d\Omega = 0_j \tag{14.3}$$

where the weights are defined to be

$$W_j(\boldsymbol{x}) \equiv H_j(\boldsymbol{x})\;, \tag{14.4}$$

the element spatial interpolation associated with node j.

The Galerkin method works well for elliptical differential equations. However, when it was applied to other classes of differential equations it was often found to yield "unstable" solutions, i.e., solutions that exhibit non-physical spatial oscillations. Generally the standard Galerkin approach is seen to break down in problems with strong boundary layer effects. The analytical approach to such problems is usually called singular perturbation theory. Typically such problems have a data dependent coefficient multiplying the highest derivative. In common special cases that coefficient approaches zero and the nature of the equation changes because of the loss of the highest derivative term. Alternatively, one can view it as a reduction in the number of boundary conditions which causes a very rapid change in the solution (i.e., a boundary layer) near the region of the "lost" boundary condition data. Some analysts divide the differential equation by

the coefficient multiplying the highest derivative term and thereby create an increased emphasis on the lower order derivative terms in the equation. One of the first studies to successfully apply a new finite element theory "for second order problems with significant first derivatives" was the use of the *Petrov-Galerkin* method by Christie, et al, in 1976 [3]. Since then the Petrov-Galerkin methods [15, 25] have generally come to be known as "stablized" formulations because they prevent the spatial oscillations and sometimes yield nodally exact solutions where the classical Galerkin method would fail badly. They are also very important because they allow equal order element interpolation for mixed nodal variables, such as pressure and velocity, that otherwise would not be possible.

14.2 Petrov-Galerkin Method

The Petrov-Galerkin method is assumed to be more general because it does not restrict the weights to just the special case of the spatial distribution of the approximating solution, but adds some additional terms to it:

$$\int_\Omega (W_j \; + \; \boldsymbol{\tau}_j \, P_j \,) \; R(\boldsymbol{x}) \; d\Omega = 0_j \tag{14.5}$$

where the P_j denotes Pertov or "stabilization" terms [9, 23]. In Eq. (14.5) the $\boldsymbol{\tau}_j$ multipiler was introduced to recognize that one would often need to account for the difference in units between W_j and P_j and to scale their relative importance in the solution. Here we will refer to each such $\boldsymbol{\tau}$ term as a "stabilization parameter" [23, 24]. If we are going to allow the weighting of the residual error to be more general than the classic Galerkin approach we are faced with selecting a rational for the additional weights. Some methods have been tried and shown to work well for some classes of equations, such as the *advective-diffusion* equation [15]. Since advection means "to carry along" it often occurs in modeling various transport phenomena. One of the common applications is heat transfer with mass flow. That is usually referred to as a convection-diffusion problem. For such problem classes the Petrov-Galerkin method is often used to create "upwind elements" as one way to stabilize the solution [2, 4, 5, 7, 10, 11, 16]. One can find many articles on those subjects, but most employ linear elements and zero source terms. These simplifications may hide more general concepts.

Consider a typical application such as convective-diffusive heat transfer governed by

$$\rho c(\frac{\partial \phi}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \phi) \; = \; \boldsymbol{\nabla}(\boldsymbol{K} \boldsymbol{\nabla} \phi) + Q \tag{14.6}$$

where $\boldsymbol{v}$ denotes a given velocity vector field, Q is the volumetric source and the material properties ρ, c, $\boldsymbol{K}$ are the mass density, specific heat and thermal conductivities, respectively. We select a generalized weight function

$$w = (\phi + p) \tag{14.7}$$

where $p(x)$ is the new Petrov or stabilization term(s). Then we invoke the method of weighted residuals:

$$\int_\Omega (\phi + p)[\rho c(\frac{\partial \phi}{\partial t} + \boldsymbol{v} \cdot \nabla \phi) - \nabla(\boldsymbol{K} \nabla \phi) - Q] d\Omega = 0. \tag{14.8}$$

Usually $\phi(x)$ is taken as continuous across element boundaries and thus allows one to employ integration by parts to yield the terms given earlier in the classical Galerkin form. The Petrov term, $p(x)$, may or may not be continuous across element boundaries. Usually it is not continuous and we can not reduce the order of the derivatives in its integrals. In either case, we can view this expanded integral form as

$$[\textit{Classical Galerkin }] + [\textit{Stablization Terms}] = 0 \tag{14.9}$$

where the stabilization term here is

$$I_s = \int_\Omega p(x)[\rho c(\frac{\partial \phi}{\partial t} + \boldsymbol{v} \cdot \nabla \phi) - \nabla(\boldsymbol{K} \nabla \phi) - Q] d\Omega \tag{14.10}$$

which is clearly zero for the exact solution. This is a typical example of a Petrov-Galerkin approach. Note that unless integration by parts can be employed this term retains the highest derivative found in the original equation. That would either increase the interpolation inter-element continuity requirement, or restrict the integral evaluation to each of the element domains rather that the full domain, Ω. The latter occurs, for example, when one includes a least squares weight (partial derivative of the residual error with respect to the unknowns) as a Petrov term [13]. This leads to a Galerkin/Least Squares (GLS) stabilization process.

It should be noted that in most low order elements the second derivatives are zero and thus the diffusion term is often omitted in the stabilization calculation. However, the second derivatives can always be estimated using patch methods or other techniques when using an iterative solution.

Since the Galerkin process works well in most cases, we review its properties and seek a change in them that may better capture advective-diffusion type solutions. The most common general form of the Petrov-Galerkin method is to pick the weights as

$$W_j = h_j + \alpha F_j \tag{14.11}$$

where the sum of the integrals of the F_j is zero. Some authors, such as Kondo et al [16], like to include additional terms in the summation in an effort to improve the numerical accuracy, but others include different residuals to provide physical insight to the stabilization terms [21].

Continuous Petrov Forms

Huebner and Thornton [12] present an example formulation where the $F_j(x)$ are picked to be continuous across the element boundaries and thus they are able to apply integration by parts, over the entire domain, to the new terms and retain the use of C° interpolation. Others have used similar approaches, such a element bubble functions, but most applications involve functions that are discontinuous across the inter-element boundaries.

Discontinuous Petrov Forms

In most advection applications behavior in the streamline direction is usually more important than in the perpendicular "cross-wind" direction. It is possible to bias the Petrov weight by defining it to be the scalar result of the dot product of a unit vector, $\boldsymbol{n}_v = -\boldsymbol{v} \,/\, \|\boldsymbol{v}\|$, in the upwind streamline direction (obtained from the velocity vector $\boldsymbol{v}$) and some other assumed vector function, say $\boldsymbol{G}(x)$:

$$p(x) = \boldsymbol{n}_v(x) \cdot \boldsymbol{G}(x) \; . \tag{14.12}$$

This common special case is know as the Streamline Upwind Petrov-Galerkin (SUPG) method [1, 10, 11, 13]. The vector function is usually taken as the gradient of the solution, $\boldsymbol{G}(x) = L\nabla\phi(x)$, where L is constant, with length dimension, introduced to keep the units consistent with those of ϕ. Since the gradient is usually discontinuous between elements, we can not use integration by parts on $p(x)$ over the solution domain.

The goal of the Streamline Upwind Petrov-Galerkin is to stabilize the solution by adding information to include a bias on gradients in the flow direction. For a given velocity $\mathbf{v}$ this is done by defining the SUPG weight function to be

$$W_j(x) = H_j(x) + \alpha h \, \nabla H_j(x) \cdot \frac{\boldsymbol{v}(x)}{\|v\|} \tag{14.13}$$

where h is a measure of the element size, and from one-dimensional studies α can be related to the local element Peclet number so as to obtain optimal accuracy [1]. In terms of the notation of Eq. (14.5), we would have $\tau_j = \alpha h / \|\boldsymbol{v}\|$, and $P_j = \mathbf{v} \cdot \Delta H_j$. Tezduyar and Osama [24] have given mathematical norm definitions for establishing both element and nodal $\boldsymbol{\tau}$ values. Other definitions of τ will be considered later.

Recall that the element interpolations based on Lagrangian methods have the property that at any point in space

$$\sum_j H_j(\boldsymbol{x}) = 1 \tag{14.14}$$

Likewise, any gradient in the x_γ spatial direction of the above sum is the null vector

$$\sum_j \frac{\partial H_j}{\partial x_\gamma}(x) = 0_\gamma \tag{14.15}$$

where typically $1 \le \gamma \le 3$. Recall here that the units have changed by the introduction of a length in the denominator due to taking a spatial derivative. For future reference consider a scalar zero term created by dotting this summation with a unit vector $\boldsymbol{n}(\boldsymbol{x})$ in the space $\boldsymbol{x}$

$$\sum_\gamma \left(\sum_j \frac{\partial N_j}{\partial x_\gamma} \right) \boldsymbol{n}_\gamma(\boldsymbol{x}) = 0 \tag{14.16}$$

Typically, we wish to consider an upwind bias and use the velocity vector to define the unit vector as

$$\boldsymbol{n}_\gamma = \frac{v_\gamma}{|v|} \tag{14.17}$$

and then the reference length h may be taken as some appropriate element distance. Finally, we multiply the result by a constant $0 \le \alpha \le 1$ to indicate the relative amount of "upwind" emphasis. From this we see that if $\boldsymbol{v}$ is constant

$$\sum_j \alpha h \nabla H_j \frac{\mathbf{v}}{\|v\|} = 0 \tag{14.18}$$

so that

$$\sum_j W_j(x) = \sum_j H_j(x) = 1 \tag{14.19}$$

as in the standard Galerkin form. Typically, α is picked to give the optimal result for a 1-D solution. That optimal value is usually defined in a collocation sense in that it exactly satisfies the PDE at a point in a uniform grid (for special choices of Q). The appearance of h in the stabilization term, of Eq. (14.13), has lead several authors to propose ways to evaluate the relevant element length to be employed. We will review some of the methods in the next section and later relate them quantitatively to other length measures related to turbulence modeling.

14.3 Geometric Measures of Element and Nodal Lengths for τ

The stabilization parameters, $\boldsymbol{\tau}$, often involve definitions that require a local length in the streamline direction related to the element size. Most researchers assume an average value over the element [1] while others allow for different lengths (and $\boldsymbol{\tau}$ values) to be associated with each node of the element [24]. We will utilize 1-D and 2-D elements to illustrate some of the available geometric constructions of h. For advection-diffusion problems formulated with 1-D linear elements it was shown that to obtain nodally exact solutions h was the element length and $Q = 0$ [1]. The same study showed an extension to quadrilaterals as illustrated in Fig. 14.2.1. Since the stabilization term was to be biased in the streamline direction it is commonly thought that the length measure should also take into consideration the flow direction. It is denoted by the unit vector **n** in the figure. There the lengths of the element, $\boldsymbol{A}$ and **B**, are established in the local coordinate directions. They are dotted with the unit vector in the streamline direction and the sum of the absolute value of the two distances is taken as the element size. A similar process can be used for linear hexahedra [1, 10], and for linear triangles [11]. These geometric approaches have been used in many stabilization studies with linear elements. For higher degree element interpolations there are fewer suggestions for geometric approaches to defining h [4, 6]. Here we will introduce some approaches for higher order Lagrangian elements in 1-D, 2-D, and 3-D. These new approaches could be extended to p-adaptive elements by using weights proportional to the number of unknowns per node.

Rather than use local coordinate directions which depend on the element type, but not its degree, one can always define geometric measures by beginning with the collection of relative position vectors from the element centroid to each of its nodes. That allows for various length projections in the streamline and cross-flow directions.

A geometric process similar to that of Brooks and Hughes [1] that establishes an element value length is shown in Fig. 14.2.2. There vectors are established at each node to create a relative nodal streamline directional distance from the element center. Those nodal distances can be employed to define a maximum element distance by using the absolute value of the most positive and negative nodal distances, as shown in 2c. An average element measure can be obtained by averaging the positive (downwind) distances

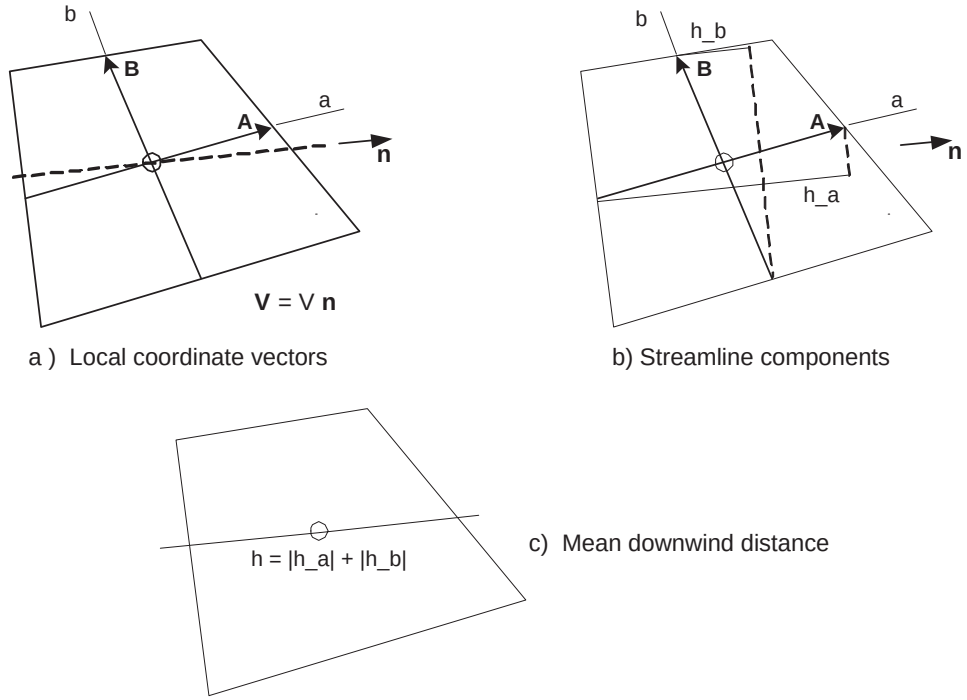
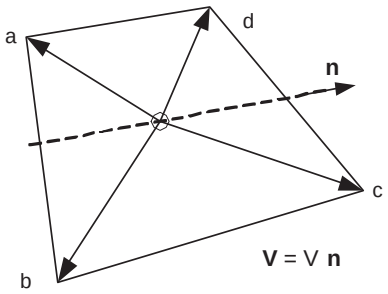


Figure 14.2.1 Classic quadrilateral element downwind distance

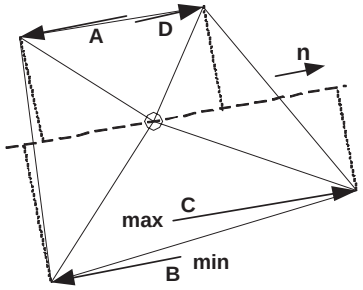
and averaging the negative (upwind) distances, as shown in 2d. Note that the number of nodes considered to be downwind (and upwind) will vary with the direction of **n**, but there will always be at least one downwind node when defined in this way. The same process works for all element types. The lengths for a linear triangular element are given in Fig. 14.2.3

Being vector based this geometric process automatically extends to 3-D space. For higher degree Lagrangian elements it allows for curvilinear shapes and indirectly accounts for the change in degree of Lagrangian elements. To illustrate these definitions for a quadratic element we begin in 1-D, in Fig. 14.2.4, where we compare linear and quadratic line elements. One can consider nodal vector lengths, in 4c, or scalar nodal distances, in 4d. For advective-diffusion in 1-D we use a stabilization parameter, $\boldsymbol{\tau}$, based on the element length, L, to obtain nodally exact solutions [1]. Codina, et al [4], conducted a similar study for 1-D quadratic elements and showed that to obtain nodally exact solutions the $\boldsymbol{\tau}$ term for the center node is approximately half that of the end nodes. For infinite Peclet numbers (pure advection) the center node $\boldsymbol{\tau}$ is exactly half the two end node $\boldsymbol{\tau}$ values. Note that in Fig. 14.2.4d the center node measure is exactly half the end node values.

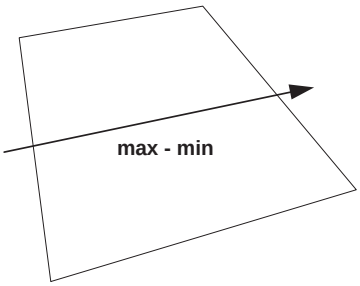
As a final geometric example for higher degree elements consider the quadratic quadrilateral element in Fig. 14.2.5. The generalization for the average downwind distance vector to the element center, **J**, always depends on the number of upwind nodes which will vary in turn with the direction of the streamline, **n**. Being based on an integer value count that upwind average can show localized jumps as **n** changes direction.



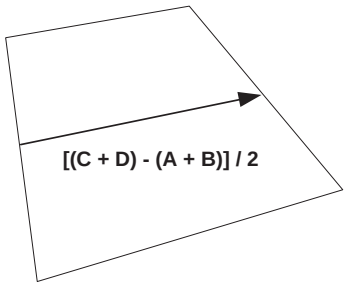
a) "Radial" position vectors



b) Nodal downwind distances, NDD

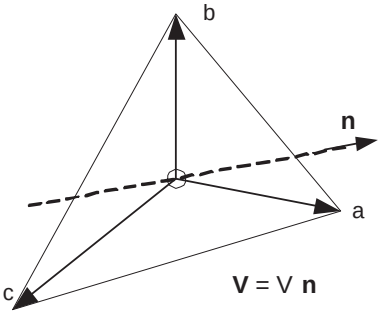


c) Maximum rule length, MD

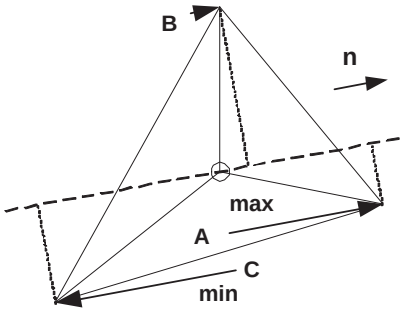


d) Average rule length, AD

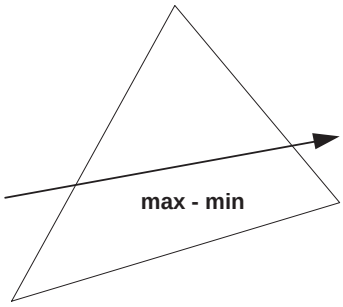
Figure 14.2.2 Quadrilateral element downwind distance options



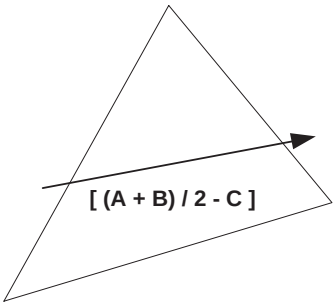
a) "Radial" position vectors



b) Nodal downwind distances, NDD



c) Maximum rule length, MD



d) Average rule length, AD

Figure 14.2.3 Triangular element downwind distance options

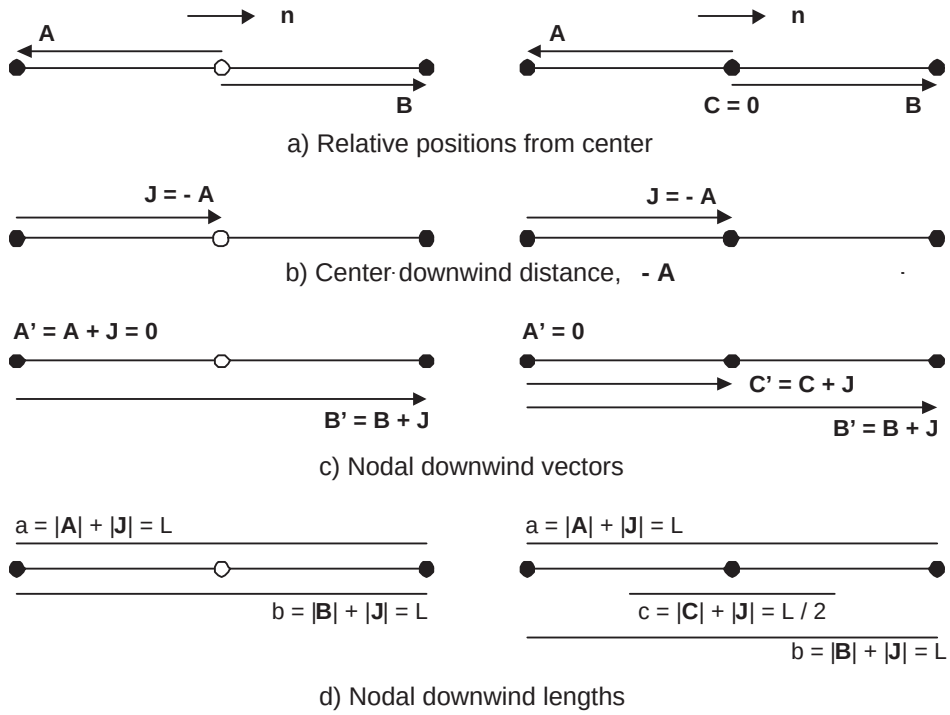


Figure 14.2.4 Linear (left) and quadratic (right) line element measures

Alternatively, the vector **J** could be taken as the negative of the largest upwind vector at the element nodes ($-$ **G** in 5b)

14.4 Review of SUPG Concepts

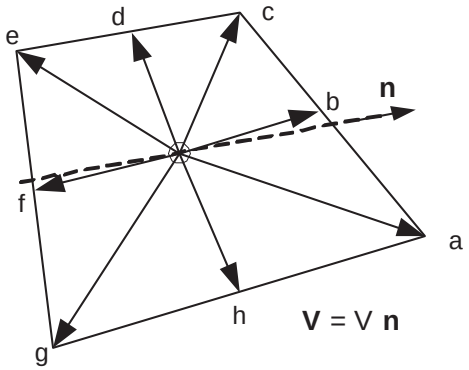
There are numerous publications on the mathematics and application of the SUPG of Brooks and Hughes [1]. Several arguments have been given to describe why the stabilization method drastically improve the results of finite solutions of non-elliptical problems. Here we will review some of the concepts but the main point of this chapter is how we implement these methods when needed. We begin our review of some of the interpretations of how these stabilization methods work with the usual approach of the one-dimensional SUPG which has been proven to exactly satisfy the homogeneous form ($Q = 0$) of Eq. (14.19) at all nodes in a uniform mesh for all Peclet numbers. Consider the one-dimensional model equation

$$u\,\frac{\partial \phi}{\partial x} - k\,\frac{\partial^2 \phi}{\partial x^2} + Q = 0, \quad x \in]\,0,\ L[\tag{14.20}$$

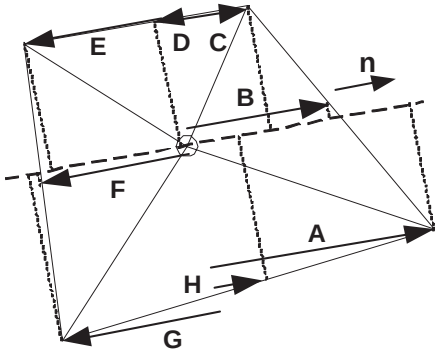
satisfying the boundary conditions of

$$\phi(0) = \phi_0\ , \quad \phi(L) = \phi_L \tag{14.21}$$

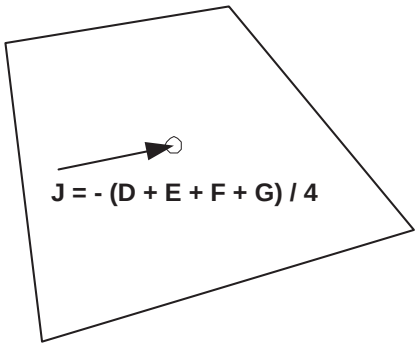
where u is the given flow velocity, k is the diffusivity coefficient, and Q is the internal source per unit length. The solution for $\phi(x)$ is governed by the global Peclet number Pe = uL/k, and the grid Peclet number, $p = uh/(2k)$, where h is the element size. For $Q = 0$



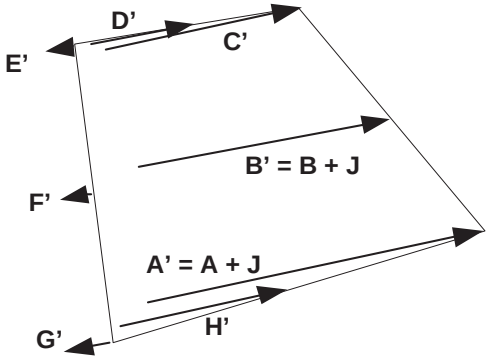
a) "Radial" position vectors



b) Nodal downwind distances



c) Average downwind to center



d) Adjusted downwind vectors

Figure 14.2.5 Assigning quadratic quadrilateral nodal downwind vectors

and u and k constant, the exact solution for $\phi_0 = 0$ and $\phi_L = 1$ is given by $\phi(x) = (1 - e^{\mathrm{Pe}\, x\,/\,L})/(1 - e^{\mathrm{Pe}})$. The classic Galerkin method solutions of this problem appear under-diffuse while most upwind methods appear over-diffuse.

Continuous Petrov Form

Only if $\boldsymbol{F}$ in Eq. 14.11 is continuous then the element matrix forms can be written, after integration by parts as the conduction or diffusion parts:

$$\boldsymbol{S}_k^e = \int_{L^e} \frac{\partial \boldsymbol{W}^T}{\partial x}\, k\, \frac{\partial \boldsymbol{H}}{\partial x}\, dx \tag{14.22}$$

$$= \int_{L^e} \frac{\partial \boldsymbol{H}^T}{\partial x}\, k\, \frac{\partial \boldsymbol{H}}{\partial x}\, dx + \alpha \int_{L^e} \frac{\partial \boldsymbol{F}}{\partial x}\, k\, \frac{\partial \boldsymbol{H}}{\partial x}\, dx \tag{14.23}$$

which matches the classical form only if $\alpha = 0$ or if $\boldsymbol{F}$ is picked to force the last integral to vanish. [12] Otherwise there will be an additional new diffusion contribution and $\boldsymbol{S}_k^e$ will usually become unsymmetric. The source resultant is

$$\boldsymbol{C}_Q^e = \int_{L^e} \boldsymbol{W}^T Q dx = \int_{L^e} \boldsymbol{H}^T Q dx + \alpha \int_{L^e} \boldsymbol{F}^T Q dx \tag{14.24}$$

which for $Q \neq 0$, modifies the nodal distribution of the source resultant. We recall that for any source distribution, $Q(x)$, the sum of all the terms in the first integral (of $\boldsymbol{H}^T Q$) accounts for the total source effects. This means that the sum of all the terms from the second integral involves $\boldsymbol{F}$, must vanish. For a constant Q that means that the sum of the $\boldsymbol{F}$ terms must vanish. If we had known fluxes on the boundary they would be coupled to $\boldsymbol{W}$ (and thus to $\alpha \boldsymbol{F}$ if it is continuous) like the volumetric source matrices were.

The new moving, or advection, contribution is the matrix

$$\boldsymbol{S}_u^e = \int_{L^e} \boldsymbol{W}^T u \frac{\partial \boldsymbol{H}}{\partial x} dx \tag{14.25}$$

which splits into

$$\boldsymbol{S}_u^e = \int_{L^e} \boldsymbol{H}^T u \frac{\partial \boldsymbol{H}}{\partial x} dx + \alpha \int_{L^e} \boldsymbol{F}^T u \frac{\partial \boldsymbol{H}}{\partial x} dx \tag{14.26}$$

which is the standard non-symmetric form plus a new array depending on $\boldsymbol{F}$ which in general would also be non-symmetric. It is much more common to employ Petrov forms that are discontinuous at the inter-element boundaries.

Discontinuous Petrov Forms

If, and only if, we consider a special case where $\boldsymbol{F}$ is proportional to the gradient of the shape functions $\boldsymbol{H}$ (say $\boldsymbol{F} = c\partial \boldsymbol{H} / \partial x$) will the new Petrov-Galerkin advection contribution due to u be a symmetrical matrix and be almost identical to the standard diffusion matrix;

$$\alpha \int_{L^e} \boldsymbol{F}^T u \frac{\partial \boldsymbol{H}}{\partial x} dx = \alpha \int_{L^e} \frac{\partial \boldsymbol{H}^T}{\partial x} cu \frac{\partial \boldsymbol{H}}{\partial x} dx. \tag{14.27}$$

Thus some people like to think of this common case as an element designed to increase the numerical diffusion in a controlled fashion.

When the SUPG is applied to this problem for linear finite elements, it gives nodally exact results by picking the optimal diffusion to add to the system. The SUPG is usually demonstrated with the finite difference pattern it produces when elements are assembled at a typical interior node of a uniform mesh [1]. Here we will take the different approach and look directly at the element matrices that result if the residual vanishes on each element. For the case of $Q = 0$;

$$\left(\frac{u}{2} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} + \frac{k}{h} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \frac{\alpha u h}{2h} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \right) \phi = 0 \tag{14.28}$$

where

$$\alpha = \text{Coth (Pe)} - 1/\text{Pe} \tag{14.29}$$

is the optimal upwind coefficient.

The first two matrices are the classical Galerkin advection and diffusion matrices, and the third square matrix is viewed as the added SUPG diffusion necessary for nodally exact solutions. While the third term is intended to emphasize the added diffusion, its units suggest that it could be added to the first matrix for the classical advection term. If we do that and apply the Petrov-Galerkin to a constant source term, Q, then we see an alternate view of the element matrices (when node 2 is downwind) is

$$\left(\frac{u}{2}\begin{bmatrix}(-1+\alpha) & (1-\alpha)\\ (-1-\alpha) & (1+\alpha)\end{bmatrix}+\frac{k}{h}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\right)\begin{Bmatrix}\phi_1\\ \phi_2\end{Bmatrix}=\frac{hQ}{2}\begin{Bmatrix}(1-\alpha)\\ 1+\alpha)\end{Bmatrix} \tag{14.30}$$

which serves as a clear reminder that the Petrov-Galerkin method also significantly influences the resultant-source vector. A system assembled from these element matrices gives the nodally exact result for any α. Two common special cases are easily observed; $\alpha \to 1$ (Pe $\to \infty$) for the maximum upwind correction

$$\left(\frac{2u}{2}\begin{bmatrix}0 & 0\\ -1 & 1\end{bmatrix}+\frac{k}{h}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\right)\begin{Bmatrix}\phi_1\\ \phi_2\end{Bmatrix}=\frac{2hQ}{2}\begin{Bmatrix}0\\ 1\end{Bmatrix} \tag{14.31}$$

This lets us think of the upwinding effects as a strong weighing the upwind rows of the element source and the advection matrices rather than modifying the diffusion. This gives some insight into why, for large Peclet numbers, the upwind method of Rice and Schnipke [18] (and its degenerate form by Shemirani and Jambunathan [20]), which deletes the upwind rows of the convection matrix, works so well for low-order multi-dimensional elements for $Q = 0$ (and Pe $\to \infty$).

However, we are interested in a general process for higher order (e.g., p-adaptive) elements, so we will consider the change from Eq. (14.27) when a quadratic line element is employed with nodes 2 and 3 being downwind. We make the common assumption that α is a single scalar term. Again we view the diffusion matrix as unchanged from the standard Galerkin form (with zero row and column sums), where node 2 is the interior node:

$$\boldsymbol{S}_k=\frac{k}{3h}\begin{bmatrix}7 & -8 & 1\\ -8 & 16 & -8\\ 1 & -8 & 7\end{bmatrix} \tag{14.32}$$

but the source vector due to Q has an additional term

$$\boldsymbol{C}_Q=\frac{Qh}{6}\begin{Bmatrix}1\\ 4\\ 1\end{Bmatrix}+\frac{\alpha QH}{6}\begin{Bmatrix}-6\\ 0\\ 6\end{Bmatrix}=\frac{Qh}{6}\begin{Bmatrix}1-6\alpha\\ 4\\ 1+6\alpha\end{Bmatrix}. \tag{14.33}$$

From this we see that the upwinding has a strong effect on the "corner" sources but no effect on the interior (and, thus, another downwind) node. Of course, the total source contributed (Qh) is still accounted for at all values of α (and Pe) since the sum of the coefficients multiplying it is unity.

The standard advection matrix for this quadratic element is the Galerkin form (with zero row sums)

$$\boldsymbol{S}_a = \frac{u}{6}\begin{bmatrix} -3 & 4 & -1 \\ -4 & 0 & 4 \\ 1 & -4 & 3 \end{bmatrix} \tag{14.34}$$

and the SUPG correction is obtained by using $k = \alpha uh/2$ in $\boldsymbol{S}_k$ (from Eq. (14.25)). The combined upwind element matrices for the SUPG method are

$$\left(\frac{u}{6}\begin{bmatrix} (7\alpha - 3) & (4 - 8\alpha) & (\alpha - 1) \\ (-4 - 8\alpha) & 16\alpha & (4 - 8\alpha) \\ (1 + \alpha) & (-4 - 8\alpha) & (3 + 7\alpha) \end{bmatrix} + \frac{k}{3h}\begin{bmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{bmatrix} \right) \phi = \frac{Qh}{6}\begin{Bmatrix} 1 - 6\alpha \\ 4 \\ 1 + 6\alpha \end{Bmatrix} \tag{14.35}$$

which is the classic Galerkin form when $\alpha = 0$. From Eq. (14.28) we see that for higher order elements, the upwinding effects on advection and source terms are not as simple as the effect of α in Eq. (14.23) may have implied. For maximum upwinding ($\alpha = 1$) this becomes

$$\left(\frac{u}{6}\begin{bmatrix} 4 & -4 & 0 \\ -12 & 16 & -4 \\ 2 & -12 & 10 \end{bmatrix} + \frac{k}{3h}\begin{bmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{bmatrix} \right) \phi = \frac{Qh}{6}\begin{Bmatrix} -5 \\ 4 \\ 7 \end{Bmatrix} \tag{14.36}$$

The above single element based α term for the quadratic element no longer gives exact results at the nodes, even for $Q = 0$. To accomplish that Codina, *et al.* [4] have shown that a nodal based approached may be needed with one upwind constant, α, for the two "corner" nodes and a second, β for the interior node. They show the constants to be

$$\begin{aligned} \beta &= (\coth (\mathrm{Pe}/2) - 2/\mathrm{Pe})/2, \\ \alpha &= \frac{(3 + 3\ \mathrm{Pe}\ \beta)\tanh(\mathrm{Pe}) - (3\mathrm{Pe} + \mathrm{Pe}^2\beta)}{(2 - 3\beta\tanh(\mathrm{Pe}))\mathrm{Pe}^2} \end{aligned} \tag{14.37}$$

which gives a β that is about half of α for most Pe, and $\beta \to \alpha/2$ for $\mathrm{Pe} \to \infty$. Their form from an element matrix viewpoint is

$$\left(\frac{u}{6}\begin{bmatrix} (7\alpha - 3) & (4 - 8\alpha) & (\alpha - 1) \\ (-4 - 8\beta) & 16\beta & (4 - 8\beta) \\ (1 + \alpha) & (-4 - 8\alpha) & (3 + 7\alpha) \end{bmatrix} + \boldsymbol{S}_k \right) \phi = \boldsymbol{C}_Q \tag{14.38}$$

which, compared to Eq. (14.35), reduces the upwind effect on the interior node. If we use the gross approximation that $\beta = \alpha/2$ (which is exact for $\mathrm{Pe} \to \infty$), we get

$$\boldsymbol{S}_a \approx \frac{u}{6}\begin{bmatrix} (7\alpha - 3) & (4 - 8\alpha) & (\alpha - 1) \\ 4(-1 - \alpha) & 8\alpha & 4(1 - \alpha) \\ (1 + \alpha) & (-4 - 8\alpha) & (3 + 7\alpha) \end{bmatrix} \tag{14.39}$$

which has zero row sums, but not zero column sums. In the limit of $\mathrm{Pe} \to \infty$, this gives

$$\boldsymbol{S}_a = \frac{u}{6}\begin{bmatrix} 4 & -4 & 0 \\ -8 & 8 & 0 \\ 2 & -12 & 10 \end{bmatrix} \tag{14.40}$$

which again significantly differs from the Galerkin form of Eq. (14.27) and the single element based α form given in Eq. (14.29). This short review of SUPG concepts suggests that the extension to higher order elements may, in general, benefit from different upwind coefficients for corner nodes, edge nodes, and internal nodes.

14.5 One-dimensional Example

The application of the SUPG for linear line elements is, as expected, the most common way to illustrate the process. There are aspects of the computation that are obtained by inspection that require little extra programming in general. It is clear that the reference length to be used in calculating the Peclet number is simply the length of the element. Likewise, the gradient of the solution, and **H**, in the x-direction is also the gradient in the direction tangent to the streamline. Finally, since the second derivatives of such approximations are zero (unless iterative results are used) one avoids having to bring into the analysis the consistent information on second derivatives that occur in the SUPG theory.

For very high Peclet numbers the governing equation needs to be modeled analytically with what is known as "singular perturbation theory". Usually such problems involve a parameter (here 1/Pe) associated with the highest order derivative in the differential equation. As that parameter approaches zero one has essentially a lower order differential equation with more boundary conditions than the reduced equation requires. In other words a thin "boundary layer" develops in a part of the solution domain near the redundant boundary condition and the solution must change very rapidly in that small region as it tries to satisfy the original higher order derivative terms. When such a problem is approximated by a classic Galerkin finite element solution a least squares spatial response develops to try to capture the very sharp gradients in thin boundary layer. While it may do that, it over shoots the spatial solution in the region adjacent to the boundary layer and gives huge errors, or physically impossible results, as it oscillates about the true solution in the main domain that is reasonably modeled by the lower order differential equation. This type of behavior should not come as a surprise since we have changed to a new class of differential equation that is unlike the elliptical one used in most of our examples. Here the system is parabolic in nature and a polynomial approximation may not be the best choice for our finite element model. Since the response is basically exponential in nature near the boundary layer we should consider using exponential interpolation functions or adding new terms to our solutions to accurately dampen out the incorrect responses. The SUPG approach does the latter. The non-polynomial interpolations would also work, but tend to be expensive to compute and sensitive to the word length of the computer employed.

We begin with a classic Galerkin solution in one-dimension with no source term and where the diffusion term, k, is decreased to make $Pe = 100$. A ten element model based on linear interpolation is shown in Fig. 14.4.1 along with the exact solution (dashed line). Is is easily seen that the effect of the sharp boundary layer has propagated back into the

full domain and gives a very inaccurate result. By way of comparison, the addition of the SUPG terms give the result in Fig. 14.4.2. The improvement is drastic, with the SUPG giving numerical results that are essentially exact at the nodes. If one reviews the error estimate for the Galerkin result (see Fig. 14.4.3) it is tempting to simply try to make the elements smaller near the boundary layer. However, that does not give much improvement (as seen in Fig. 14.4.4) and a much finer mesh would be required to attempt to get reasonable accuracy and thus the SUPG modifications are much more cost effective.

The one-dimensional source code, in Fig. 14.4.5, illustrates typical considerations of SUPG methods. This version is designed to let the student switch from the standard Galerkin to SUPG by supplying the keyword *supg* in the input file. In the one-dimensional case we know that the fluid velocity is constant and acts over the full length of the element. Thus, one can select to compute the upwind parameters outside the element numerical integration loop. They are illustrated in lines 46 to 51. Other changes that relate to the SUPG selection occur between lines 70 to 91. The consistent SUPG method introduces second derivates of the interpolation functions. They may or may not be zero. Lines 72 through 78 address their inclusion, as do lines 84 and 85, even though most programmers choose to omit them. If we neglect the second derivative question we always have to append new matrices to the element source vector and square matrix. The main difference in the element matrices, lines 89 through 91, compared top previous examples is that we have both the interpolations for **W** and **H** appearing in the matrix products instead of just **H**. Note that the physical derivatives of **H** in the x-direction, $DGH(1,:)$ is actually a derivative taken tangent to the fluid streamline so the term $u * DGH(1,:)$ in line 91 is related to the speed of the flow times the gradient of the unknown along the streamline. Also remember that line 91 makes the square matrix non-symmetric. The data file for this example is in Fig. 14.4.6. The exact solution to be compared with (case 18) is identified in line 3 while its use is invoked in lines 19 and 20. It requires the global Peclet number and that is supplied as miscellaneous data as the last line (52) and is not used anywhere in the application source code of Fig. 14.4.5.

14.6 Generalizing to Higher Dimensions

Here we will outline the generalization of the previous process to a single implementation that can handle 1-D, 2-D, 3-D, or axisymmetric domains for any element in the MODEL library. Generalizing the SUPG method requires much more data to describe the velocity field and required items along the streamline directions. The current version allows the choice of four different definitions of τ. Thus the example program is quite a bit longer but is easily broken into four conceptual tasks.

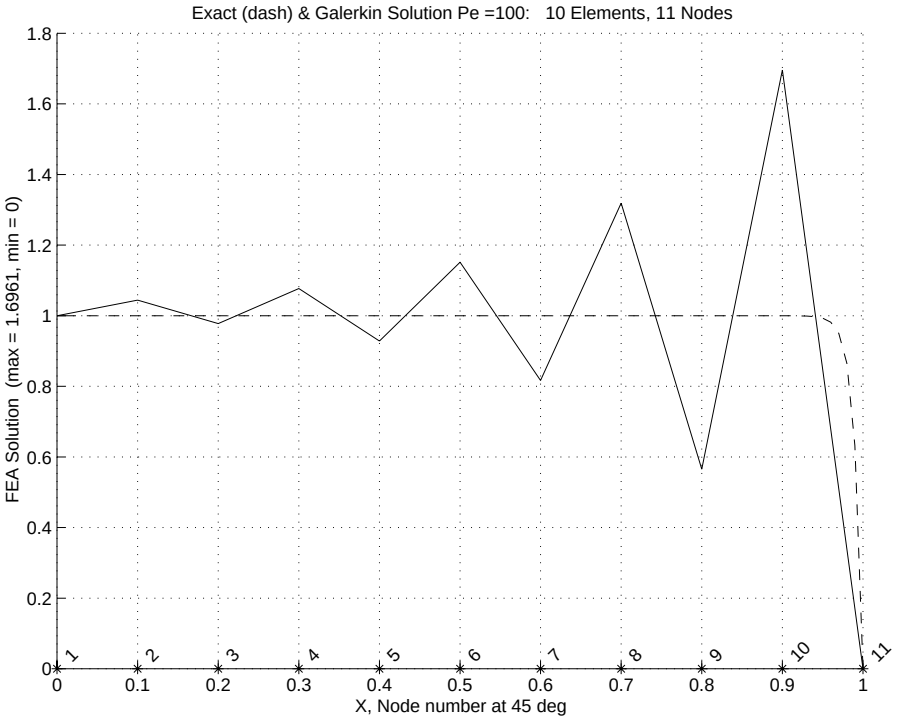


Figure 14.4.1 Galerkin solution for Pe = 100

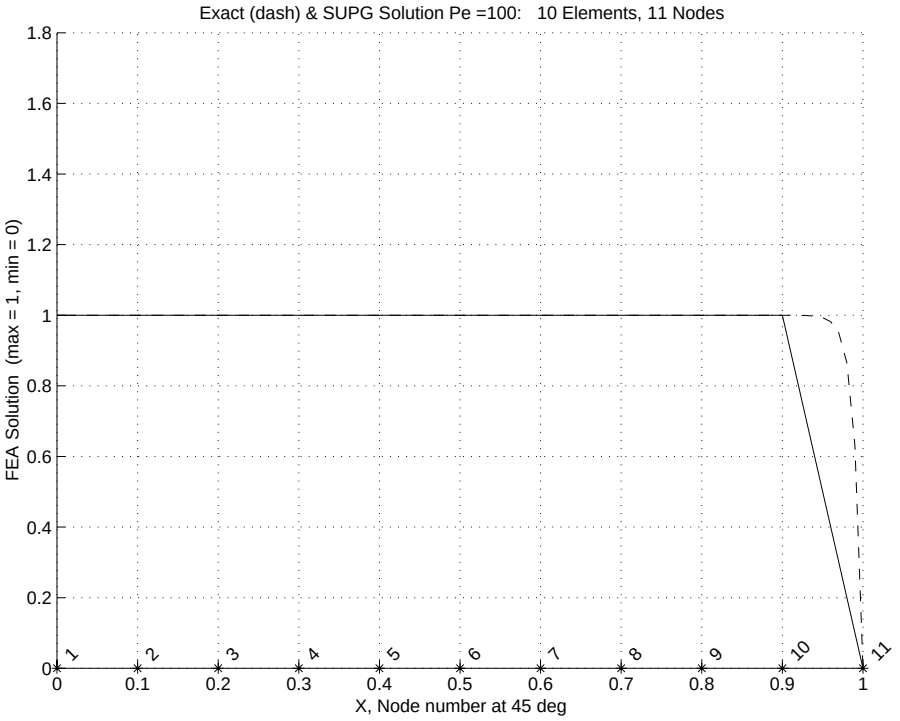


Figure 14.4.2 SUPG solution for Pe = 100

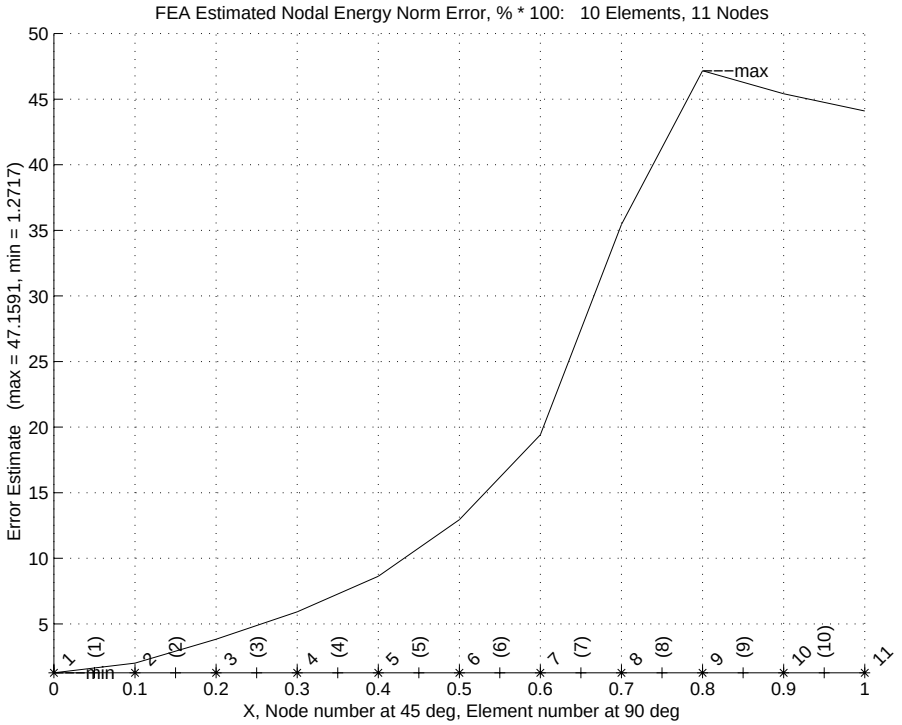


Figure 14.4.3 Galerkin energy error norm estimate

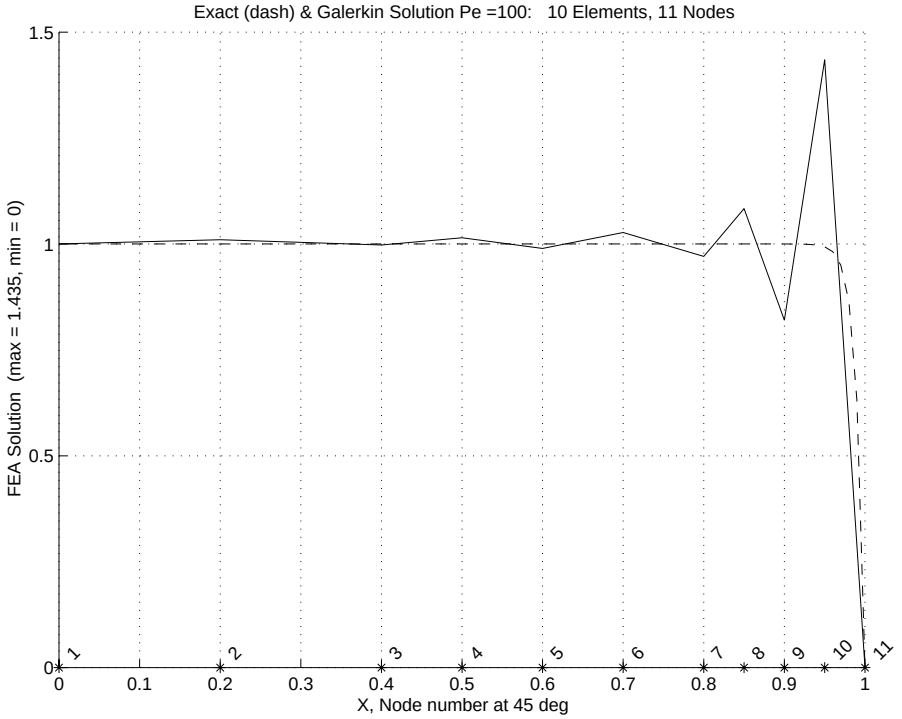


Figure 14.4.4 Revised mesh Galerkin solution for Pe = 100


```
[  1] ! .....................................................
[  2] ! ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***
[  3] ! .....................................................
[  4] ! Define any new array or variable types, then give statements
[  5] !          (Library example number 112)
[  6] !     Galerkin or  SUPG 1-d Advection-Diffusion Problem
[  7] !  u * dp/dx - d(k * dp/dx)/dx = Q, assume u, k, Q constant
[  8] !
[  9] ! u      = GET_REAL_LP (1)      ! velocity
[ 10] ! k      = GET_REAL_LP (2)      ! conductivity
[ 11] ! Q      = GET_REAL_LP (3)      ! source per unit length
[ 12] ! SUPG   = global logical flag  ! F=Galerkin (default), T=SUPG
[ 13]
[ 14] ! LT_N     = number of nodes for this element type
[ 15] ! MISC_FX  = number of integer miscellaneous properties
[ 16] ! N_LP_FLO = number of real element properties
[ 17] ! W        = Petrov weight, DGW its global derivative
[ 18]
[ 19]   REAL(DP) :: W (LT_N), DGW (1, LT_N)  ! SUPG & deriv
[ 20]   REAL(DP) :: D2GH (1, LT_N)           ! SUPG, zero ?
[ 21]   REAL(DP) :: DL, DX_DR, DL_A          ! Length, Jacobian
[ 22]   REAL(DP) :: u, k, Q                  ! input data
[ 23]   REAL(DP) :: Pe, ALPHA, COTH          ! Peclet data, L2
[ 24]   INTEGER  :: IQ                       ! Loops
[ 25]   REAL(DP), SAVE :: Pe_max             ! debugging
[ 26]
[ 27]
[ 28]   DL    = COORD (LT_N, 1) - COORD (1, 1) ! Element length
[ 29]   DX_DR = DL / 2.                        ! constant Jacobian
[ 30]   DL_A  = DL / (LT_N - 1)                ! SUPG length
[ 31]
[ 32] !   DATA READS AND SAVES
[ 33]   u = GET_REAL_LP (1)            ! velocity
[ 34]   k = GET_REAL_LP (2) ; E = k    ! conductivity
[ 35]   Q = 0.d0 ; IF ( N_LP_FLO > 2 ) Q = GET_REAL_LP (3) ! source
[ 36]   E = k                          ! constitutive
[ 37]
[ 38]   IF ( IE == 1 ) THEN ! FIRST ELEMENT, ONE TIME ACTIONS
[ 39]     Pe_max = 0.d0                                  ! initialize
[ 40]
[ 41]     IF ( .NOT. SUPG ) THEN                         ! echo choice
[ 42]            PRINT *,'NOTE: Galerkin method'         ! default
[ 43]     ELSE ; PRINT *,'NOTE: SUPG method' ; END IF    ! supg
[ 44]   END IF ! FIRST ELEMENT
[ 45]
[ 46] !    SUPG TERMS (ASSUMING L2 ELEMENT),   ? Bias for L3, L4 ?
[ 47]   Pe   = ABS(u) * DL / k                 ! Grid Peclet
[ 48]   IF ( Pe > Pe_max )  Pe_max = Pe        ! for debug
[ 49]   COTH  = COSH (Pe/2) / SINH (Pe/2)      ! Optimal SUPG
[ 50]   ALPHA = ABS (COTH) - 1.d0 / ABS (Pe/2) ! Optimal SUPG L2
[ 51]   ALPHA = SIGN (ALPHA, u)                ! abs(ALPHA)*sign of u
```

Figure 14.4.5(cont.) One-dimensional SUPG source code


```
[ 52]
[ 53] !      ------- ELEMENT MATRICES FORMATION ----------
[ 54]   CALL STORE_FLUX_POINT_COUNT ! Save LT_QP FOR SCP
[ 55]
[ 56]   DO IQ = 1, LT_QP    ! LOOP OVER QUADRATURES, S, C zeroed
[ 57]
[ 58] !        GET TRIAL INTERPOLATION FUNCTIONS, AND X-COORD
[ 59]     H    = GET_H_AT_QP (IQ)   ! SOLUTION INTERPOLATION
[ 60]     XYZ  = MATMUL (H, COORD)  ! ISOPARAMETRIC
[ 61]
[ 62] !        LOCAL AND GLOBAL FIRST DERIVATIVES
[ 63]     DLH   = GET_DLH_AT_QP (IQ) ! LOCAL DERIVATIVE
[ 64]     DGH   = DLH / DX_DR        ! PHYSICAL DERIVATIVE
[ 65]
[ 66] !  *** SELECT STANDARD GALERKIN OR SUPG ***
[ 67]     IF ( .NOT. SUPG ) THEN  ! Galerkin
[ 68]       W = H ; DGW (1, :) = DGH (1, :)
[ 69]
[ 70]     ELSE ! SUPG Method
[ 71] !        LOCAL AND GLOBAL SECOND DERIVATIVES (FOR N_SPACE == 1)
[ 72]       SELECT CASE (LT_N)     ! ELEMENT LIBRARY CHECK
[ 73]         CASE (2) ; D2LH = 0.d0
[ 74]         CASE (3) ; CALL DERIV2_3_L (PT (1, IQ), D2LH (1, :))
[ 75]         CASE (4) ; CALL DERIV2_4_L (PT (1, IQ), D2LH (1, :))
[ 76]         CASE DEFAULT ; STOP 'NO SECOND DERIVATIVE IN LIBRARY'
[ 77]       END SELECT
[ 78]       D2GH = D2LH / DX_DR**2  ! PHYSICAL SECOND DERIVATIVE
[ 79]
[ 80] !        SUPG WEIGHTINGS, NOTE SECOND DERIVATIVE IN DGW
[ 81]       W          = H          + ALPHA * DGH  (1, :)*DL_A*0.5d0
[ 82]       DGW (1, :) = DGH (1, :) + ALPHA * D2GH (1, :)*DL_A*0.5d0
[ 83] !        PRE-INSERT SECOND DERIVATIVE RESIDUAL, IF ANY
[ 84]       IF ( LT_N > 2 ) S = S + k * ALPHA * DL_A * WT (IQ)      &
[ 85]                 * DX_DR * OUTER_PRODUCT (DGH (1, :), D2GH (1, :))
[ 86]     END IF ! Method option
[ 87]
[ 88] !          MATRICES: SOURCE, CONDUCTION & ADVECTION
[ 89]     C = C + Q * W * WT (IQ) * DX_DR    ! SOURCE
[ 90]     S = S + ( k * MATMUL (TRANSPOSE(DGH), DGH)              &
[ 91]         + u * OUTER_PRODUCT (W, DGH(1,:) )) * WT (IQ) * DX_DR
[ 92]
[ 93] !-->  SAVE COORDS, E, DERIVATIVE MATRIX FOR POST PROCESSING
[ 94]     CALL STORE_FLUX_POINT_DATA (XYZ, E, DGH)
[ 95]   END DO  ! QUADRATURE
[ 96]   IF ( IE == N_ELEMS ) PRINT *,'Maximum element Pe = ', PE_max
[ 97] !    ***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***
```

Figure 14.4.5 One-dimensional SUPG source code

For higher dimensional problems we can form a single element based upwind length measure by evaluating the velocity at the element center to form nodal distances, such as in Figs. 14.2.2b and 3b, and then convert them to a single length as illustrated in Figs. 14.2.2c and 3c. Denote that length as h_{geom}. Along the streamline we could use the 1-D optimal length scaling, α_{opt}, based on the local element Peclet number (at the center) to compute a corresponding stabilization term:

$$\tau_{geom} = \alpha_{opt}\, h_{geom} \,/\, 2\,\|\mathbf{u}\|\,.$$


```
[ 1] title "SUPG Advection-Diffusion, Pe=100"
[ 2] example     112 ! Application source code library number
[ 3] exact_case   18 ! Analytic solution for list_exact, etc
[ 4] bar_chart       ! Include bar chart printing in output
[ 5] b_rows        1 ! Number of rows in the B (operator) matrix
[ 6] dof           1 ! Number of unknowns per node
[ 7] el_nodes      2 ! Maximum number of nodes per element
[ 8] elems        10 ! Number of elements in the system
[ 9] gauss         4 ! Maximum number of quadrature points
[10] nodes        11 ! Number of nodes in the mesh
[11] line_el         ! Major elements are line elements
[12] space         1 ! Solution space dimension
[13] el_homo         ! Element properties are homogeneous
[14] el_real       3 ! Number of real properties per element
[15] supg            ! Use streamline upwind Petrov-Galerkin method
[16] reals         1 ! Number of miscellaneous real properties
[17] pt_list         ! List the answers at each node point
[18] remarks       5 ! Number of user remarks, e.g. property names
[19] list_exact      ! List given exact answers at nodes, etc
[20] list_exact_flux ! List given exact fluxes at nodes, etc
[21] unsymmetric     ! Unsymmetric skyline storage is used
[22] end             ! Terminate control, remarks follow
[23] 1 u * dp/dx - d(k * dp/dx)/dx = Q, assume u, k, Q constant
[24] 2 for  Pe * u,x + u,xx = 0, u(0) = 1, u(1) = 0, Pe=u/k, Q=0
[25] 3 Exact u(x) = (EXP(Pe * X) - EXP (Pe))/(1.d0 - EXP (Pe))
[26] 4 For Pe >> 1 we loose u,xx and the second required EBC
[27] 5 except for a small boundary layer near that EBC
[28]   1  1  0.  ! node, bc_flag, x
[29]   2  0  0.1
[30]   3  0  0.2
[31]   4  0  0.3
[32]   5  0  0.4
[33]   6  0  0.5
[34]   7  0  0.6
[35]   8  0  0.7
[36]   9  0  0.8
[37]  10  0  0.9
[38]  11  1  1.0 ! end nodes
[39]   1   1   2 ! elem, n1, n2
[40]   2   2   3
[41]   3  3  4
[42]   4  4  5
[43]   5  5  6
[44]   6  6  7
[45]   7  7  8
[46]   8  8  9
[47]   9  9 10
[48]  10 10 11  ! end elements
[49]   1  1 1.0 ! node, dof, bc_value
[50]  11  1 0.0 ! node, dof, bc_value
[51]   1 1.0 0.01   0.0 ! u, k, Q
[52]   100. ! Misc real: Pe for exact solution use
```

Figure 14.4.6 Data for 1-D SUPG test

Those same nodal distances can also be averaged over the element volume and the total number of nodes by using their interpolations over the element to define another upwind length, h_{vol}:

$$h_{vol} = \frac{\sum_j \int_{L^e} H_j \, h_j}{\sum_j \int_{L^e} H_j}$$

which likewise defines a stabilization parameter τ_{vol}. These two geometric definitions of element lengths and τ's will be extended with two additional definitions here that come from more mathematical justifications. Tezduyar and Park [22] defined an alternate geometric length. It is known as h_{ugn} since it comes from the dot product of **u** and the gradient of the generalized element interpolation functions,**N**. For our scalar variable examples it is:

$$h_{ugn} = 2\,\|\mathbf{u}\| \left(\sum_j |\, \mathbf{u} \cdot \nabla H_j \,| \right)^{-1} .$$

They define the corresponding advection dominated flow stabilization parameter to be

$$\tau_{ugn} = h_{ugn} \,/\, 2\,\|\mathbf{u}\|$$

and a similar form for stabilizing the least squares incompressiblity constraint in Navier-Stokes flows.

More recently Tezduyar and Osama [24] suggested a τ parameter based on scaling the Galerkin and stabilization matrices to be of the same order in each element. Thus they use the ratios of two matrix norms to establish the stabilization parameters for advection, diffusion, and transient dominated regions. Since the resulting τ involves the ratio of the norms of two matrices it has been found to be relatively insensitive to the method chosen to evaluate a matrix norm. Here we will use the norm to be the square root of the sum of the squares of all the terms in the matrix.

Note from Eq. 14.8 that the Galerkin contribution will produce three square matrices. They come from transient, advection, and diffusion dominated terms. A discontinuous Petrov stabilization term would also contribute similar terms but usually linear elements are employed so the second derivative contribution is zero in that case. The setting of τ by a matrix norm method is an attempt to assure that the Galerkin and Pertov terms are of the same order of magnitude, relative to the effect that is dominating the flow. For advective dominated flow the definition of the τ_{norm} given by the ratio norms of the two matrices arising from the $\boldsymbol{v} \cdot \nabla \phi$ terms:

$$\tau_{norm} = \frac{\| \int_{L^e} \boldsymbol{H}^T \boldsymbol{v} \cdot \nabla H \, dx \|}{\| \int_{L^e} \boldsymbol{v} \cdot \nabla H^T \, \boldsymbol{v} \cdot \nabla H \, dx \|}$$


```
[  1]! ..............................................................
[  2]!    ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***
[  3]! ..............................................................
[  4]!  Advection-Diffusion Equations: 1-D, 2-D, 3-D, Axisymmetric
[  5]
[  6]!          u * Del P - Del ( E Del P) + r P - Q = 0
[  7]!              VIA NUMERICALLY INTEGRATED ELEMENTS
[  8]
[  9]! MISCELLANEOUS REAL PROPERTIES: (1) = diffusity
[ 10]! (2) = SOURCE, Q, (optional, defaults to 0)
[ 11]! (3) = r, (optional, defaults to 0)
[ 12]! (4) = THICKNESS  (optional, defaults to 1 or radius)
[ 13]! NOTE: u is defined via subroutine VELOCITY_AT_POINT
[ 14]
[ 15] REAL(DP) :: CONST, DET, DET_WT, THICK ! integration
[ 16] REAL(DP) :: SOURCE, RATE              ! data: Q & r
[ 17] INTEGER  :: IP                        ! counter
[ 18]
[ 19]!  Required upwind items
[ 20]  REAL(DP) :: CENTER (N_SPACE)           ! average of nodes
[ 21]  REAL(DP) :: U      (N_SPACE)           ! Velocity vector
[ 22]  REAL(DP) :: UNIT_V (N_SPACE), SPEED    ! unit vector, speed
[ 23]  REAL(DP) :: U_DGH  (LT_N)              ! streamline gradient
[ 24]  REAL(DP) :: D2GH   (N_2_DER, LT_N)     ! 2nd deriv of H
[ 25]  REAL(DP) :: E_UP, E_CROSS              ! Diffusion up & cross
[ 26]  REAL(DP) :: VISCOSITY                  ! in E
[ 27]  REAL(DP) :: TAU                        ! stabilize term
[ 28]
[ 29]!  Stabilization matrix notations
[ 30]  REAL(DP) :: S_M     (LT_FREE, LT_FREE)  ! SUPG sq matrix
[ 31]  REAL(DP) :: S_C     (LT_FREE, LT_FREE)  ! SUPG sq matrix
[ 32]  REAL(DP) :: S_K     (LT_FREE, LT_FREE)  ! SUPG sq matrix
[ 33]  REAL(DP) :: S_K_BAR (LT_FREE, LT_FREE)  ! SUPG sq matrix
[ 34]  REAL(DP) :: S_R_BAR (LT_FREE, LT_FREE)  ! SUPG sq matrix
[ 35]  REAL(DP) :: S_UP    (LT_FREE, LT_FREE)  ! SUPG sq matrix
[ 36]  REAL(DP) :: C_UP    (LT_FREE)           ! SUPG column matrix
[ 37]
[ 38]!  Optional geometric upwind items
[ 39]  REAL(DP) :: RADIAL (LT_N, N_SPACE)     ! relative positions
[ 40]  REAL(DP) :: DOWN   (LT_N)              ! downwind wrt center
[ 41]  REAL(DP) :: DOWN_NODAL (LT_N)          ! downwind total
[ 42]  REAL(DP) :: GEOM_H,   VOL_H            ! element DW lengths
[ 43]  REAL(DP) :: GEOM_TAU, VOL_TAU          ! element Tau values
[ 44]  REAL(DP) :: PECLET, ALPHA              ! Re Peclet number
[ 45]  LOGICAL  :: IS_DOWNWIND (LT_N)         ! true if downwind node
[ 46]
[ 47]!  Optional norm based upwind items
[ 48]  REAL(DP) :: ONE_PT (LT_PARM), ONE_WT   ! 1 pt rule
[ 49]  REAL(DP) :: UGN_TAU                    ! ugn
[ 50]  REAL(DP) :: S1_TAU, NORM_C, NORM_K_BAR ! norms
[ 51]
[ 52]  S_M = 0 ; S_C = 0 ; S_K = 0 !  Initialize element
[ 53]
```

Figure 14.5.1a Storage for general advection-diffusion


```
[ 54]!-->  DEFINE ELEMENT PROPERTIES
[ 55]  RATE = 0 ; SOURCE = 0 ; THICK = 1 ; VISCOSITY = 1 ! initialize
[ 56]  IF ( REALS > 0 ) VISCOSITY = GET_REAL_MISC (1) ! constant diffusity
[ 57]  IF ( REALS > 1 ) SOURCE    = GET_REAL_MISC (2) ! constant Q
[ 58]  IF ( REALS > 2 ) RATE      = GET_REAL_MISC (3) ! constant r
[ 59]  IF ( REALS > 3 ) THICK     = GET_REAL_MISC (4) ! constant thickness
[ 60]
[ 61]  CENTER = SUM ( COORD, DIM=1 ) / LT_N           ! center point
[ 62]  CALL APPLICATION_E_MATRIX (IE, CENTER, E)      ! constitutive law
[ 63]
[ 64]  IF ( SUPG ) THEN ! Streamline Upwind Petrov-Galerkin additions
[ 65]
[ 66]!      INITIALIZE STABILIZATION ARRAYS
[ 67]    S_UP = 0.d0 ; C_UP = 0.d0 ; H_INTG = 0.d0
[ 68]    S_K_BAR = 0.d0 ; S_R_BAR = 0.d0
[ 69]
[ 70]!      GET CENTER VELOCITY AND DIFFUSION
[ 71]    CALL VELOCITY_AT_POINT (CENTER, U, UNIT_V, SPEED)  ! velocity
[ 72]
[ 73]!       GET DIFFUSION ALONG STREAMLINE
[ 74]    CALL DIFFUSION_UPWIND (E, UNIT_V, E_UP, E_CROSS)   ! transform
[ 75]
[ 76]    IF ( TAU_GEOM .OR. TAU_VOL ) THEN
[ 77]      CALL GET_RADIALS_FROM_CENTER (CENTER, RADIAL)
[ 78]      CALL GET_DOWNWIND_LOGIC (RADIAL, UNIT_V, DOWN, IS_DOWNWIND)
[ 79]      CALL GET_MAX_DOWNWIND_DIST (DOWN, DOWN_NODAL)
[ 80]    END IF
[ 81]
[ 82]    IF ( TAU_GEOM ) THEN
[ 83]      GEOM_H = ABS (MINVAL (DOWN)) + MAXVAL (DOWN)
[ 84]      PECLET = 0.5d0 * SPEED * GEOM_H / E_UP
[ 85]      CALL PECLET_OPTIMAL_RULE (PECLET, ALPHA)
[ 86]      GEOM_TAU = 0.5d0 * GEOM_H * ALPHA / SPEED
[ 87]    END IF ! Tau_geom
[ 88]
[ 89]    IF ( TAU_UGN ) THEN
[ 90]      CALL GET_ONE_PT_RULE (ONE_PT, ONE_WT)           ! local point
[ 91]      CALL SCALAR_DERIVS (ONE_PT, DLH)                ! deriv of H
[ 92]      AJ  = MATMUL (DLH, COORD)                       ! Jacobian, J
[ 93]      CALL INVERT_JACOBIAN (AJ, AJ_INV, DET, N_SPACE) ! Inverse of J
[ 94]      DGH     = MATMUL (AJ_INV, DLH)                  ! Del H
[ 95]      U_DGH   = MATMUL (U, DGH)                       ! u dot Del H
[ 96]      UGN_TAU = 1.d0 / SUM ( ABS (U_DGH) )            ! Tau ugn value
[ 97]    END IF ! Tau_ugn
[ 98]
[ 99]  END IF ! Initialize upwinding
[100]
[101]!      STORE NUMBER OF POINTS FOR FLUX CALCULATIONS
[102]  CALL STORE_FLUX_POINT_COUNT ! Save LT_QP
[103]
```

Figure 14.6.1b Computations at element center


```
[104]!-->   NUMERICAL INTEGRATION LOOP
[105]   DO IP = 1, LT_QP
[106]     H   = GET_H_AT_QP (IP)    ! EVALUATE INTERPOLATION FUNCTIONS
[107]     XYZ = MATMUL (H, COORD)   ! FIND GLOBAL COORD, ISOPARAMETRIC
[108]     DLH = GET_DLH_AT_QP (IP)  ! FIND LOCAL DERIVATIVES
[109]     AJ  = MATMUL (DLH, COORD) ! FIND JACOBIAN AT THE PT
[110]
[111]!         FORM INVERSE AND DETERMINATE OF JACOBIAN
[112]     CALL INVERT_JACOBIAN (AJ, AJ_INV, DET, N_SPACE)
[113]     IF ( AXISYMMETRIC ) THICK = TWO_PI * XYZ (1) ! via axisymmetric
[114]     CONST  = DET * WT(IP) * THICK                ! local measure
[115]     H_INTG = H_INTG + H * CONST                  ! H integral
[116]
[117]!         EVALUATE GLOBAL DERIVATIVES, DGH == B
[118]     DGH = MATMUL (AJ_INV, DLH)         ! Physical gradient H
[119]!    Note: D2GH assumed zero here       ! 2nd Derivs H
[120]     B   = DGH                          ! copy DGH into B
[121]
[122]!      VARIABLE VOLUMETRIC SOURCE, via keyword use_exact_source
[123]!      Defaults to file my_exact_source_inc if no exact_case key
[124]     IF ( USE_EXACT_SOURCE )  CALL        & ! analytic Q
[125]          SELECT_EXACT_SOURCE (XYZ, SOURCE) ! via exact_case key
[126]
[127]!      GALERKIN SOURCE TERM
[128]     C = C + CONST * SOURCE * H         ! source resultant
[129]
[130]!      DIFFUSION SQUARE MATRIX
[131]     S_K = S_K + CONST * MATMUL ((MATMUL (TRANSPOSE (B), E)), B)
[132]
[133]!      ADD RATE SQUARE MATRIX from -r*U
[134]     S_M = S_M + RATE * OUTER_PRODUCT (H, H) * CONST
[135]
[136]!      IGNORE SQUARE MATRIX FROM 2nd DERIVATIVES, INITIALLY
[137]
[138]!         SET STREAMLINE DIRECTION (AND DEFAULT IF SPEED = 0)
[139]     CALL VELOCITY_AT_POINT (XYZ, U, UNIT_V, SPEED)
[140]
[141]!       ADVECTION SQUARE MATRIX -V*Grad_U
[142]     U_DGH = MATMUL (U, DGH)                        ! vel dot grad H
[143]     S_C   = S_C + OUTER_PRODUCT (H, U_DGH) * CONST ! no upwind
[144]
[145]     IF ( SUPG ) THEN ! UPWIND AT QP
[146]
[147]!         GET DIFFUSION ALONG STREAMLINE, E_UP, FROM E TENSOR
[148]       CALL DIFFUSION_UPWIND (E, UNIT_V, E_UP, E_CROSS)
[149]
[150]!          FORM STABILIZATION ARRAYS (LESS Tau SCALE)
[151]       C_UP    = C_UP    + SOURCE * U_DGH   * CONST
[152]       S_K_BAR = S_K_BAR + OUTER_PRODUCT (U_DGH, U_DGH) * CONST
[153]       !             - 2nd deriv, & variable E, now neglected
[154]       S_R_BAR = S_R_BAR + OUTER_PRODUCT (U_DGH, H) * RATE * CONST
[155]     END IF ! SUPG VARIABLE UPWIND
[156]
[157]!-->  SAVE COORDS, E AND DERIVATIVE MATRIX, FOR POST PROCESSING
[158]     CALL STORE_FLUX_POINT_DATA (XYZ, (E * THICK), B)
[159]
[160]   END DO ! for integration
```

Figure 14.6.1c Numerical integration of advection-diffusion items


```
[161]
[162]    S = S_K + S_M + S_C  ! if no upwinding
[163]
[164]    IF ( TAU_VOL ) THEN ! integral average downwind dist
[165]      VOL_H = DOT_PRODUCT (H_INTG, DOWN_NODAL) &
[166]            / SUM (H_INTG) / LT_N    ! integral average
[167]      PECLET = 0.5d0 * SPEED * VOL_H / E_UP
[168]      CALL PECLET_OPTIMAL_RULE (PECLET, ALPHA)
[169]      VOL_TAU = 0.5d0 * VOL_H * ALPHA / SPEED
[170]    END IF ! Tau geom
[171]
[172]    IF ( SUPG ) THEN !     STABILIZE SOLUTION, DEFAULT TO S1
[173]      NORM_C     = SQRT ( SUM ( S_C **2     ) ) ! 2 norm
[174]      NORM_K_BAR = SQRT ( SUM ( S_K_BAR **2 ) ) ! 2 norm
[175]      S1_TAU     = NORM_C / NORM_K_BAR          ! norm method
[176]
[177]      IF    ( TAU_GEOM ) THEN   ! keywords supg and tau_geom
[178]        TAU = GEOM_TAU
[179]      ELSEIF ( TAU_UGN  ) THEN  ! keywords supg and tau_ugn
[180]        TAU = UGN_TAU
[181]      ELSEIF ( TAU_VOL  ) THEN  ! keywords supg and tau_vol
[182]        TAU = VOL_TAU
[183]      ELSEIF ( TAU_S1   ) THEN  ! keywords supg and tau_norm
[184]        TAU = S1_TAU
[185]      ELSE                      ! keyword supg only
[186]        TAU = S1_TAU
[187]      END IF ! user selection
[188]
[189]!     FORM SUPG ADDITIONS FOR AN ELEMENT BASED TAU
[190]      C    = C + C_UP * TAU
[191]      S_UP = (S_K_BAR + S_R_BAR) * TAU
[192]      S    = S +  S_UP
[193]    END IF ! SUPG
[194]!     ***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***
```

Figure 14.6.1d Final selection of τ and SUPG stabilizations

```
# UPWIND_WORD     ! REMARKS                                    [DEFAULT]
supg              ! Use streamline upwind Petrov-Galerkin method [F]
tau_geom          ! use Akin geometry method for SUPG Tau        [F]
tau_norm          ! use Tezduyar norm method for SUPG Tau        [T]
tau_ugn           ! use Tezduyar UGN method for SUPG Tau         [F]
tau_vol           ! use Akin volume method for SUPG Tau          [F]
```

Figure 14.6.2 New control options for stabilized solutions

14.7 Two-dimensional Examples

The extension of SUPG methods two higher dimensions is relatively clear but there are choices to be made on the most cost effective way to get the effective Peclet number. This may involve elements where the velocity is clearly constant in a element, or it may have significant changes over an element (as in a high degree p-method formulation). Some logical options will be considered after some typical numerical results are presented. We will begin using only the classic linear triangular element (T3) and consider higher degree elements in later examples. A common test case is where a fluid enters the lower left edge of a rectangle and exits at the lower right edge as shown in

Fig. 14.7.1. The boundary condition on the unknown is that it varies rapidly from zero to two along the inflow boundary and is zero at the impervious sides. Along the inflow edge (of the negative x-axis) the given value is $T = 1 + \tanh((2 * x + 1) * 10)$ for $x = [-1, 0]$. The outflow and interior values are to be determined. For an infinite Peclet number (no diffusion) the outflow values should be the mirror images of the input curve. The velocity components are $u = 2y(1 - x^2)$ and $v = -2x(1 - y^2)$ which means that for relatively large elements both the magnitude and direction of the velocity may change significantly within the element. This velocity field is maximum at the origin, zero on three sides, and is clockwise about the origin.

An initially uniform mesh of linear, T3, triangular elements is selected as shown in Fig. 14.7.1 along with essential boundary condition flags at the nodes, and a typical low velocity Galerkin solution. That mesh was designed so that the same nodes can be used to form a similar mesh made with quadratic, T6, triangles, or the corresponding Q4 or Q9 quadrilateral elements. When the local Peclet numbers are low a reasonable Galerkin solution can be obtained without stabilization, as illustrated in Fig. 14.7.1. But if one increases the maximum Peclet number the a unstable Galerkin solution results as seen in Fig. 14.7.2. Retaining the same data but stabilizing the solution (simply by adding control keyword *supg* to the input) renders a drastically improved solution whose front and back views are seen in Figs. 14.7.3. respectively. Applying the four stabilization choices gives the inlet and outlet solutions (for $y = 0$), and the solution profiles alone the mid-plane ($x = 0$) as shoen in Figs. 14.7.4 and 5, respectively.

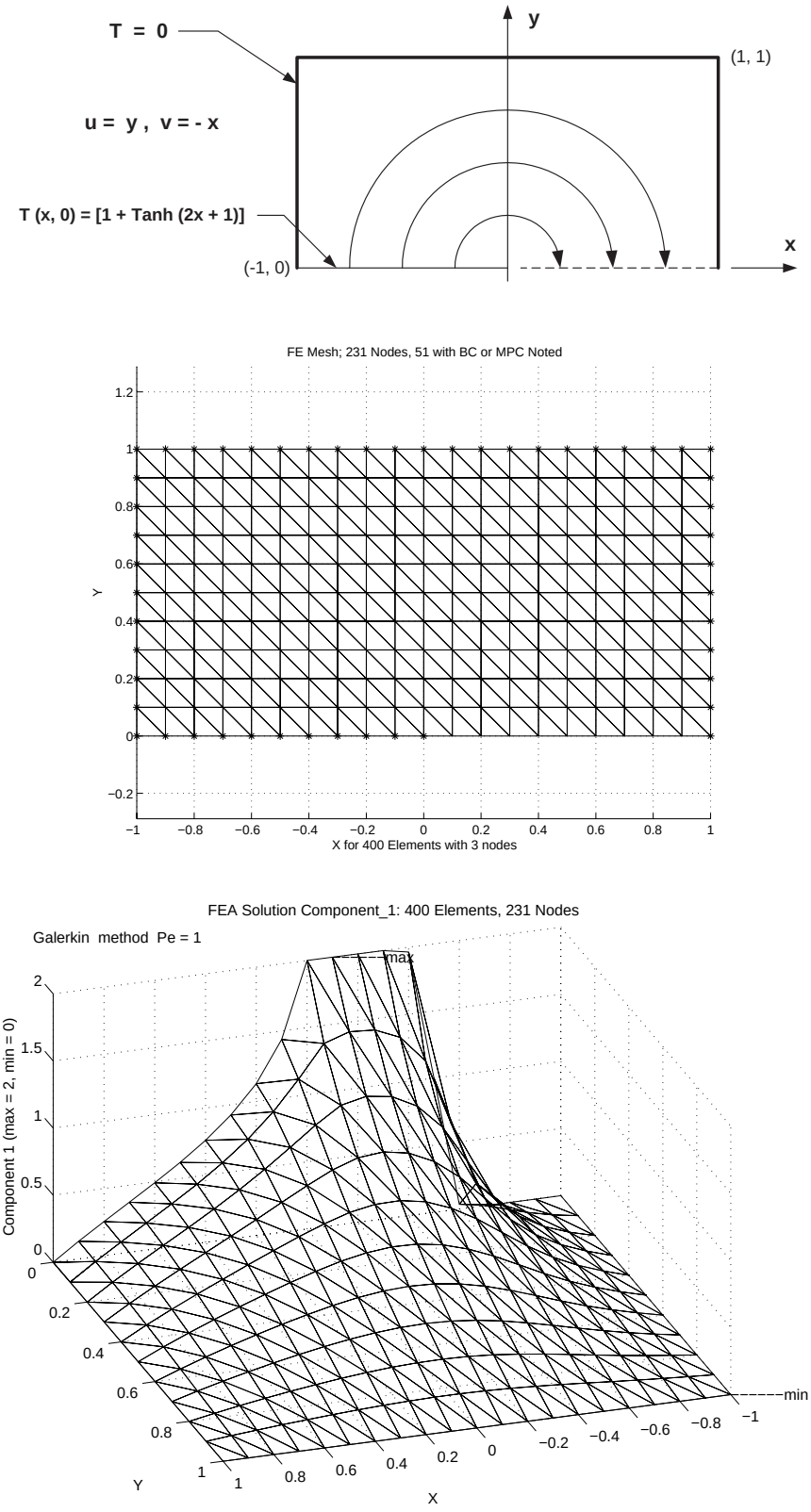


Figure 14.7.1 Smith-Hutton test, initial T3 mesh, and low Pe Galerkin solution

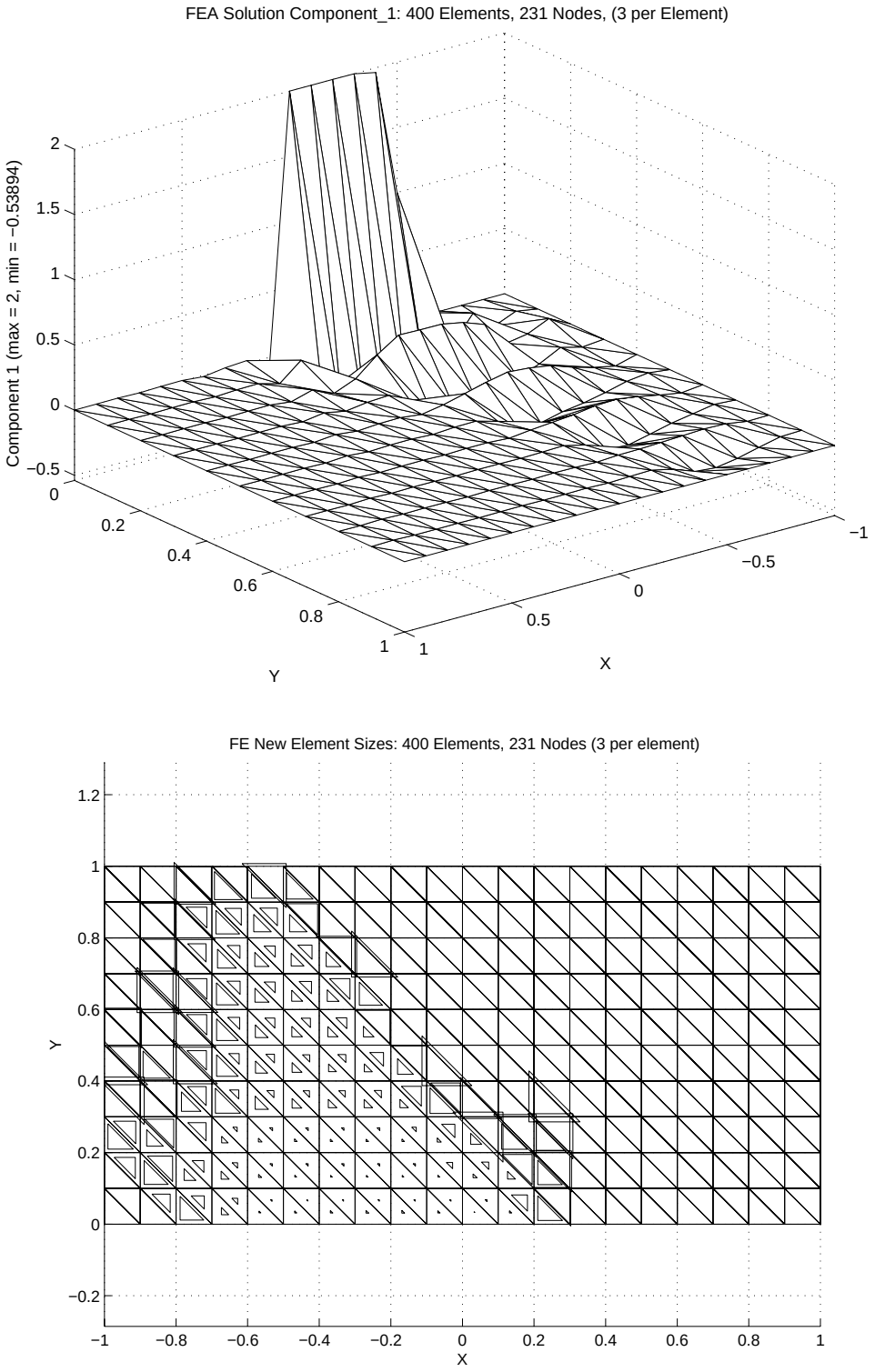


Figure 14.7.2 A very high Pe Galerkin T3 solution and suggested mesh

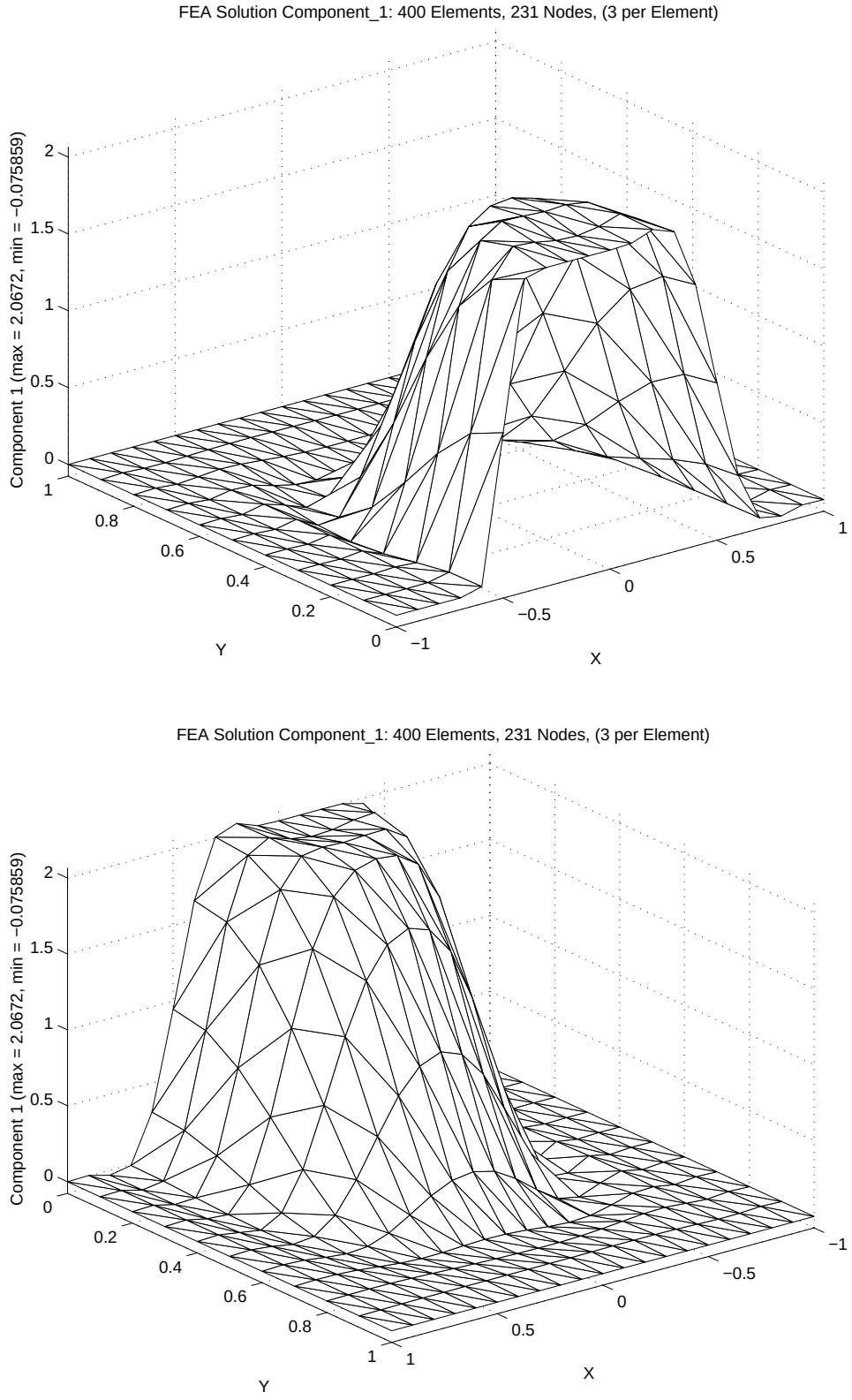


Figure 14.7.3 A very high Pe SUPG T3 solution (front and back)

Figure 14.7.4 Initial inlet ($x \le 0$) and outlet values for $y = 0$

Figure 14.7.5 Initial mid-plane ($x = 0$) values

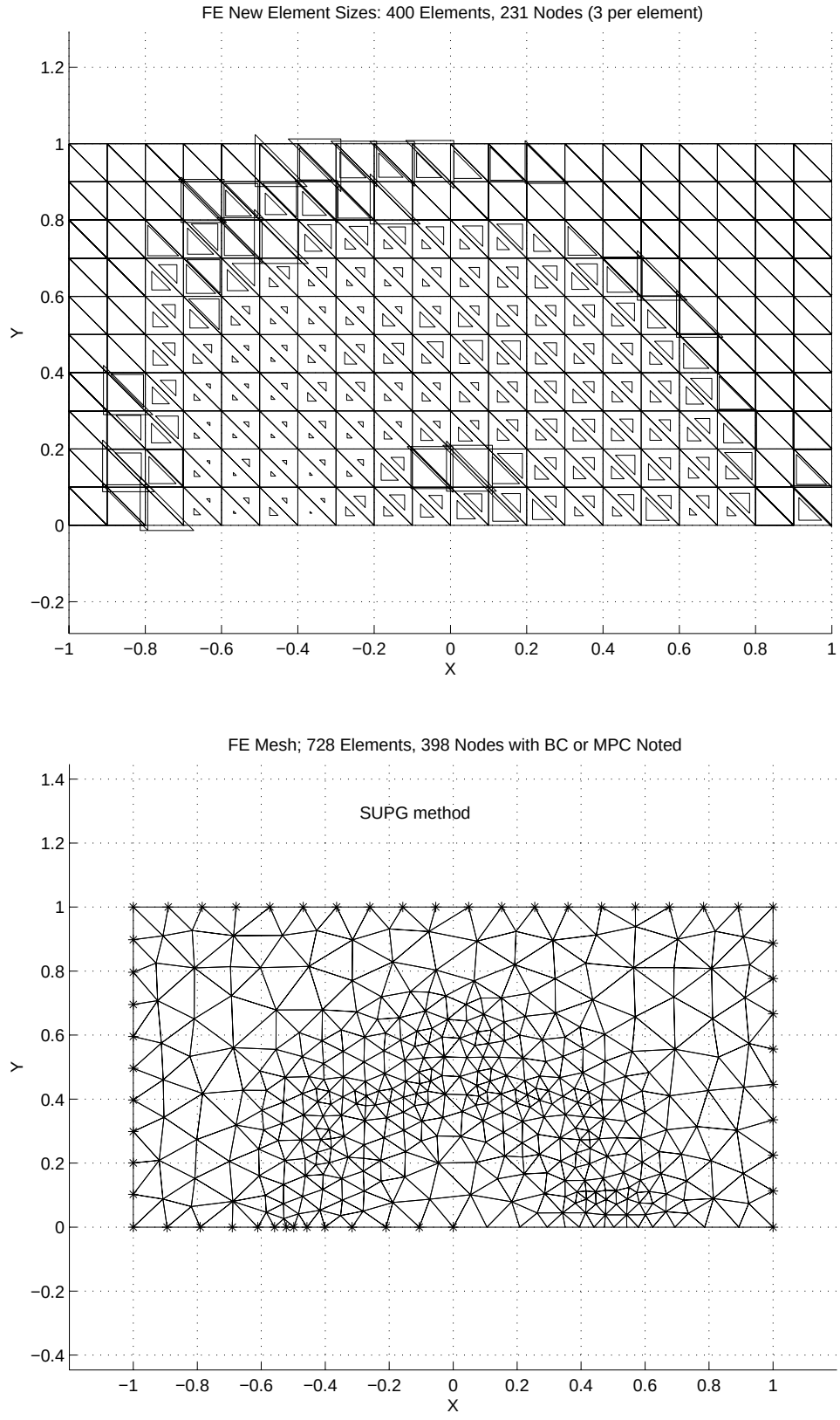


Figure 14.7.6 Estimated sizes and revised SUPG T3 mesh

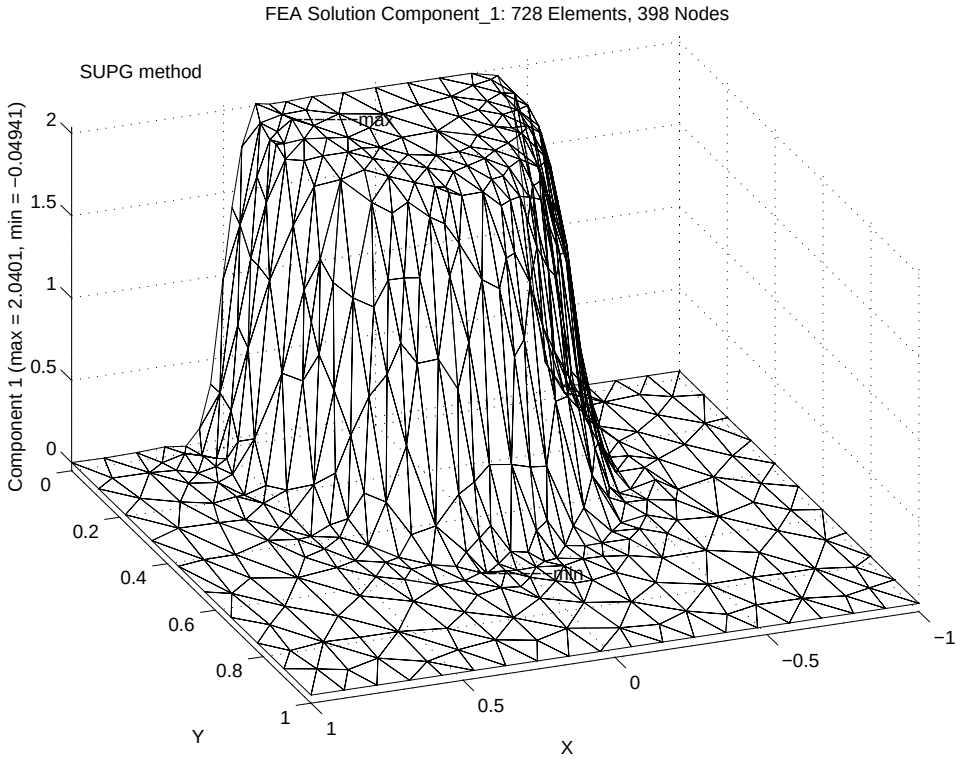


Figure 14.7.7 Revised SUPG T3 solution with τ_{norm}

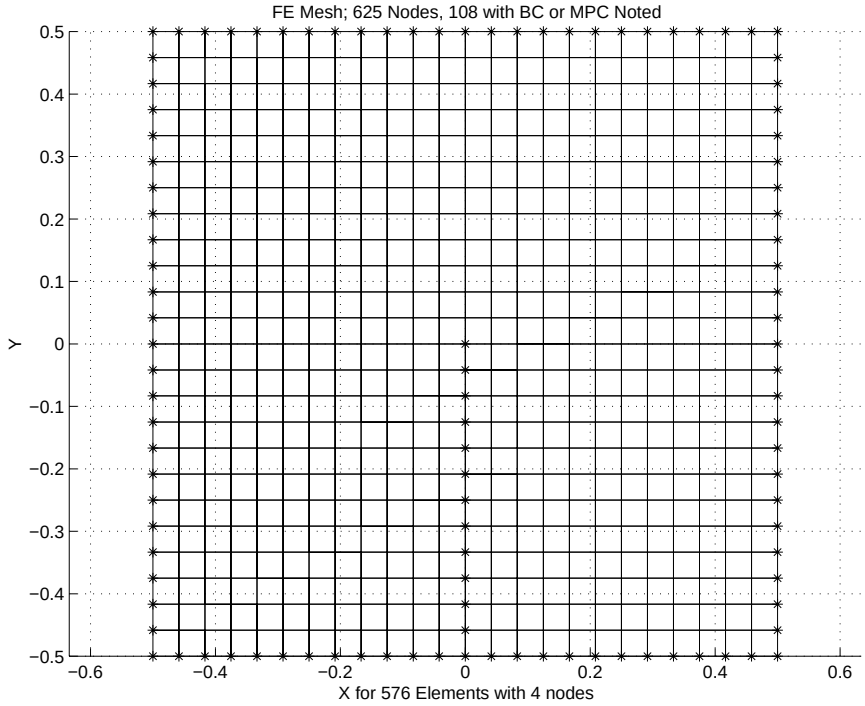


Figure 14.7.8 The Cosine Hill initial Q4 mesh

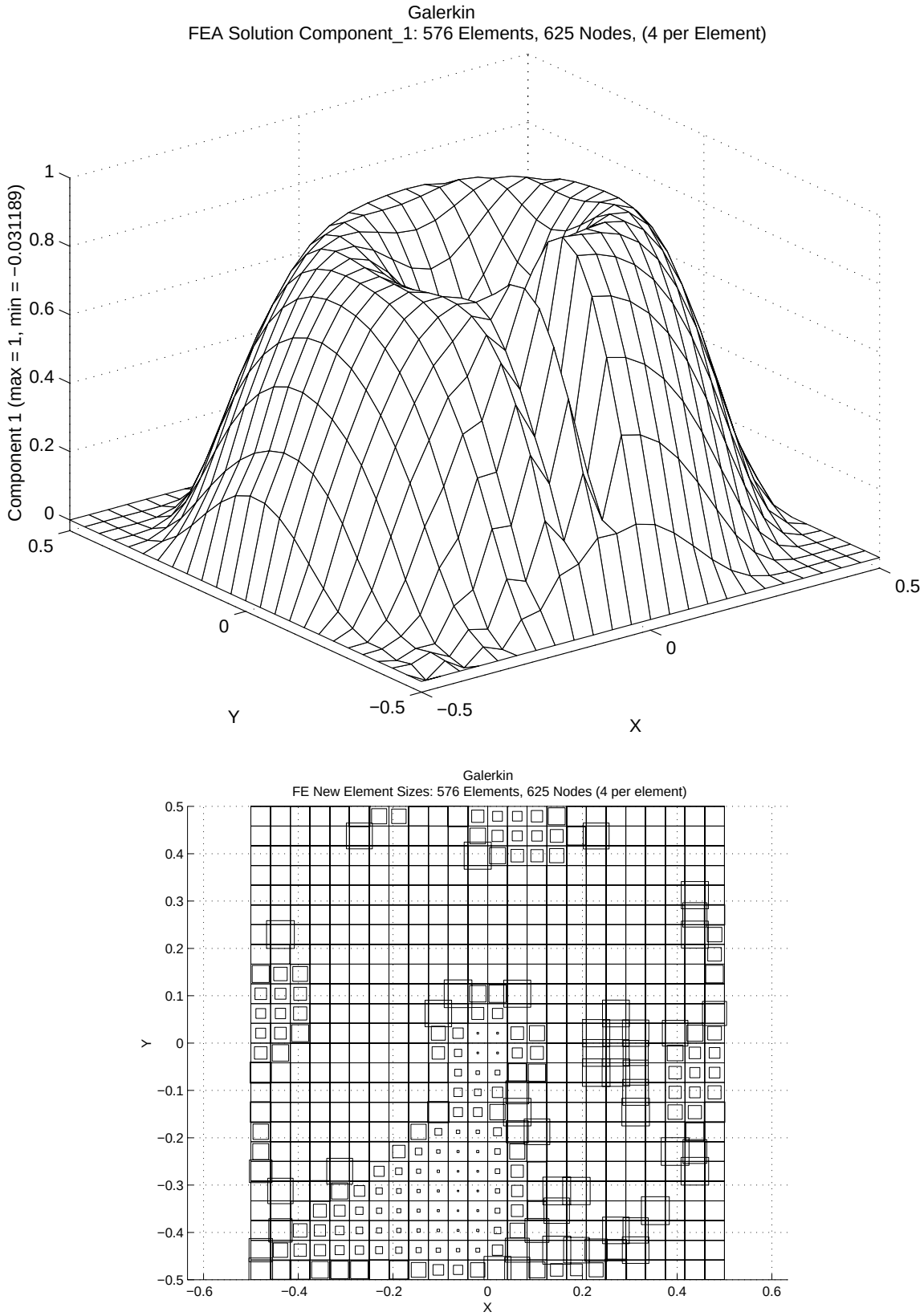


Figure 14.7.9 Typical Galerkin Q4 solution and mesh refinement suggestion

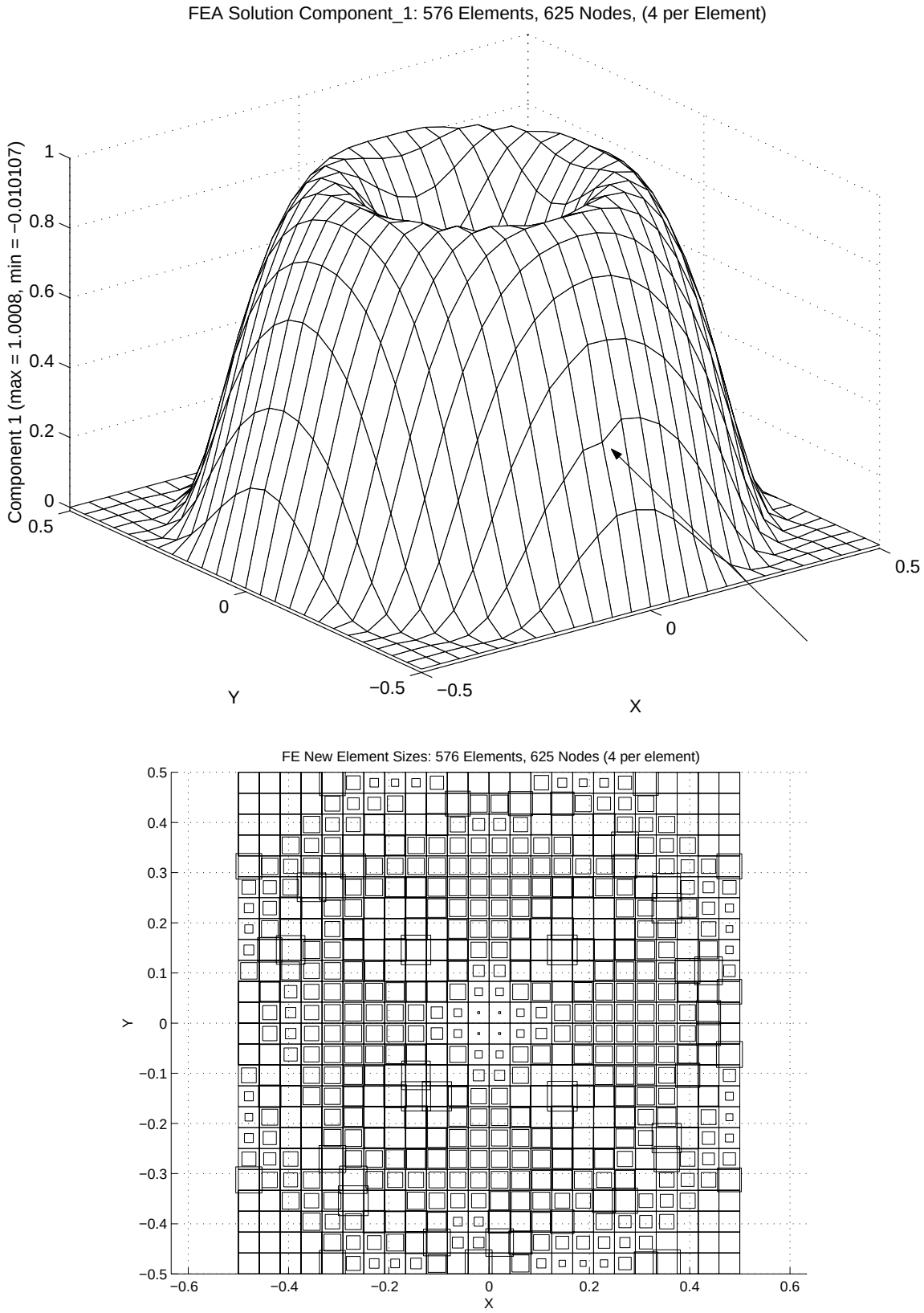


Figure 14.7.10 The Q4 result and new mesh estimate from τ_{vol}

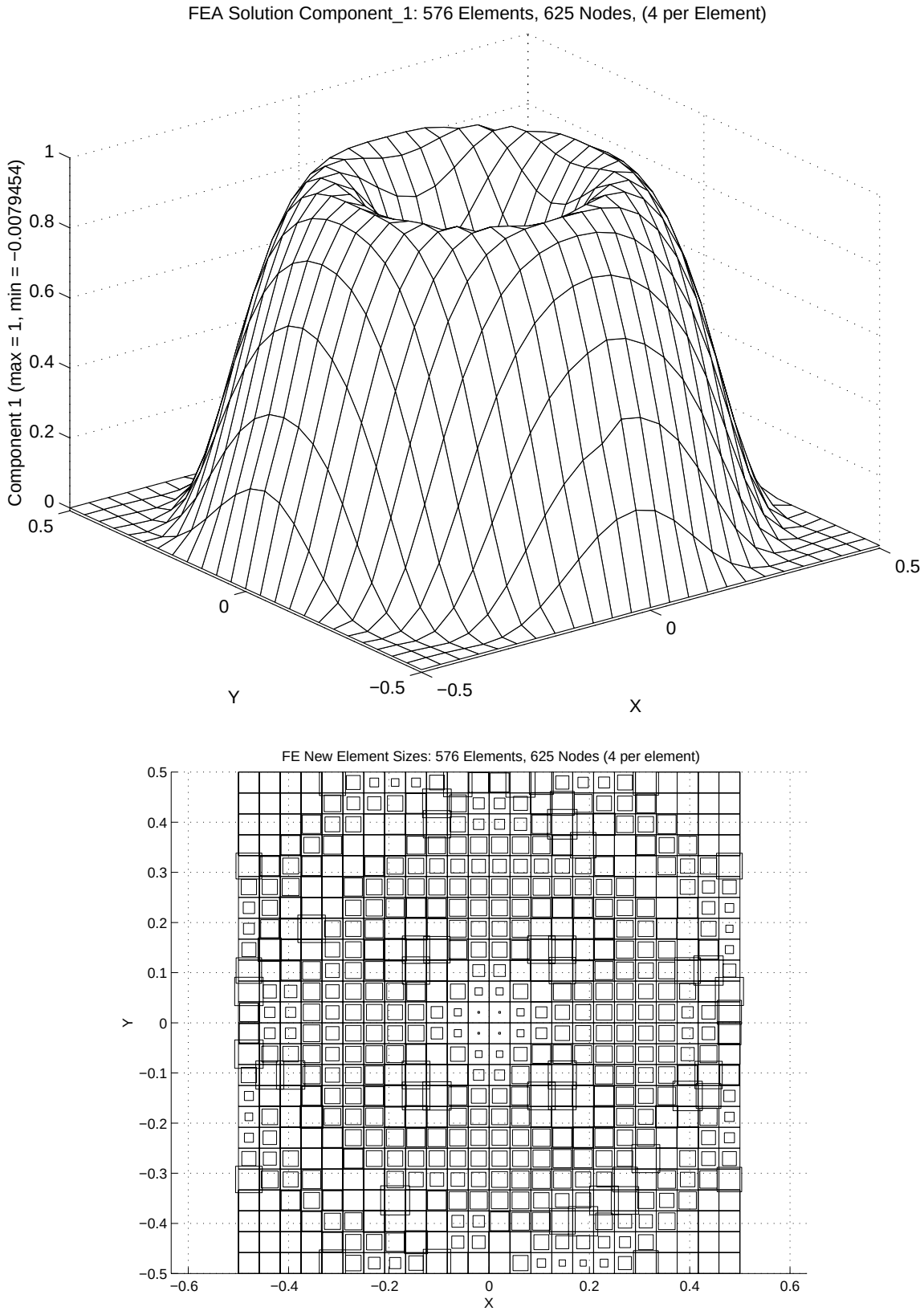


Figure 14.7.11 The Q4 result and new mesh estimate from τ_{geom}

Employing the element error estimators suggests that the current element sizes should be scaled as shown in Fig.14.7.6. Supplying those guidelines for element densities to an automatic mesh generator yield a second mesh, in the same figure, and whose new solution surface is displayed in Fig. 14.7.7. This is a typical example of how stabilized solutions are essential for non-elliptical differential equations.

As a second stabilization example consider a common test problem called the Cosine Hill. It assumes almost pure advection ($k = 1e-8$) using a counterclockwise circular velocity field centered on a square, with $-1/2 \le x, y \le 1/2$. The fluid speed proportional to the radial distance from the center of the square. As shown in Fig. 14.7.8, the initial mesh has $\phi = 0$ on the boundary of the square. On the interior line at $x = 0$ and $y \le 0$ it varies as $\phi = \text{Sin}\,(\pi(1-2y))$. In this case the source term, Q, is zero. For pure advection, the solution surface should circle back on its self with no change. That is, in the limit we should see a zero value of $\partial\phi/\partial x$ as the solution approaches $x = 0$ from $x < 0$. Typical Galerkin solutions would have very large oscillations as they approach $x = 0$.

The initial Q4 element mesh, in Fig. 14.7.8, has been chosen so that it is uniform and so that without changing the nodal count or locations one can employ a mesh for Q4, Q9, Q16, T3, T6, or T10 elements. This allows one to compare linear, quadratic, and cubic elements. In addition, all of these meshes yield nodal results that can be projected to a common Q4 mesh for visual comparisons. This mesh can be refine uniformly to retain this feature, or if one uses an error estimator new non-uniform meshs can be developed. The results depend on the element degree, the choice of τ, and the relative element sizes and locations. Thus, it is well suited for parametric studies of those variables. Many such studies have been carried out but space here limits us to a few examples.

A typical Galerkin solution result is shown in Fig.14.7.9 along with the estimated required mesh refinement. One could continually refine the mesh in this fashion and possibly obtain a useful Galerkin solution, but it is more more cost effective to employ a stabilization method.

Figure 14.7.10 shows an initial solution using Q4 elements and the τ_{vol} choice. The arrow indicates a lack of smoothness common to most of the solutions. Note that the vertical axis of these plots gives the minimum and maximum function value found anywhere in the mesh. That allows some extra comparisons of the various surface plots to follow which at first glance look very similar when projected onto a common reference surface. This solution yielded error estimates that suggested a new non-uniform mesh with local element sizes also shown there. Note that the new sizes are not symmetrically spaced and refine the region near $x = 0^-$ more. The corresponding figures for τ_{geom}, τ_{ugn}, and τ_{norm}, using the Q4 elements, are given in Figs. 14.7.11 to 13, respectively.

We will illustrate the other linear through cubic elements, that can use the same nodes, by applying the τ_{norm} to the initial mesh only. We begin with the Q9 and Q16 elements. The Q9 solution results are shown in Fig. 14.7.14 and those for the Q16 elements are in Fig. 14.7.15. These are cude meshes even though the polynomial degree is higher. The corresponding first suggested mesh revisions are given in Fig. 14.7.16, but one probably needs even smaller sizes (which can be done via control keywords). The corresponding linear through cubic T3, T6, and T10 triangular family element results and

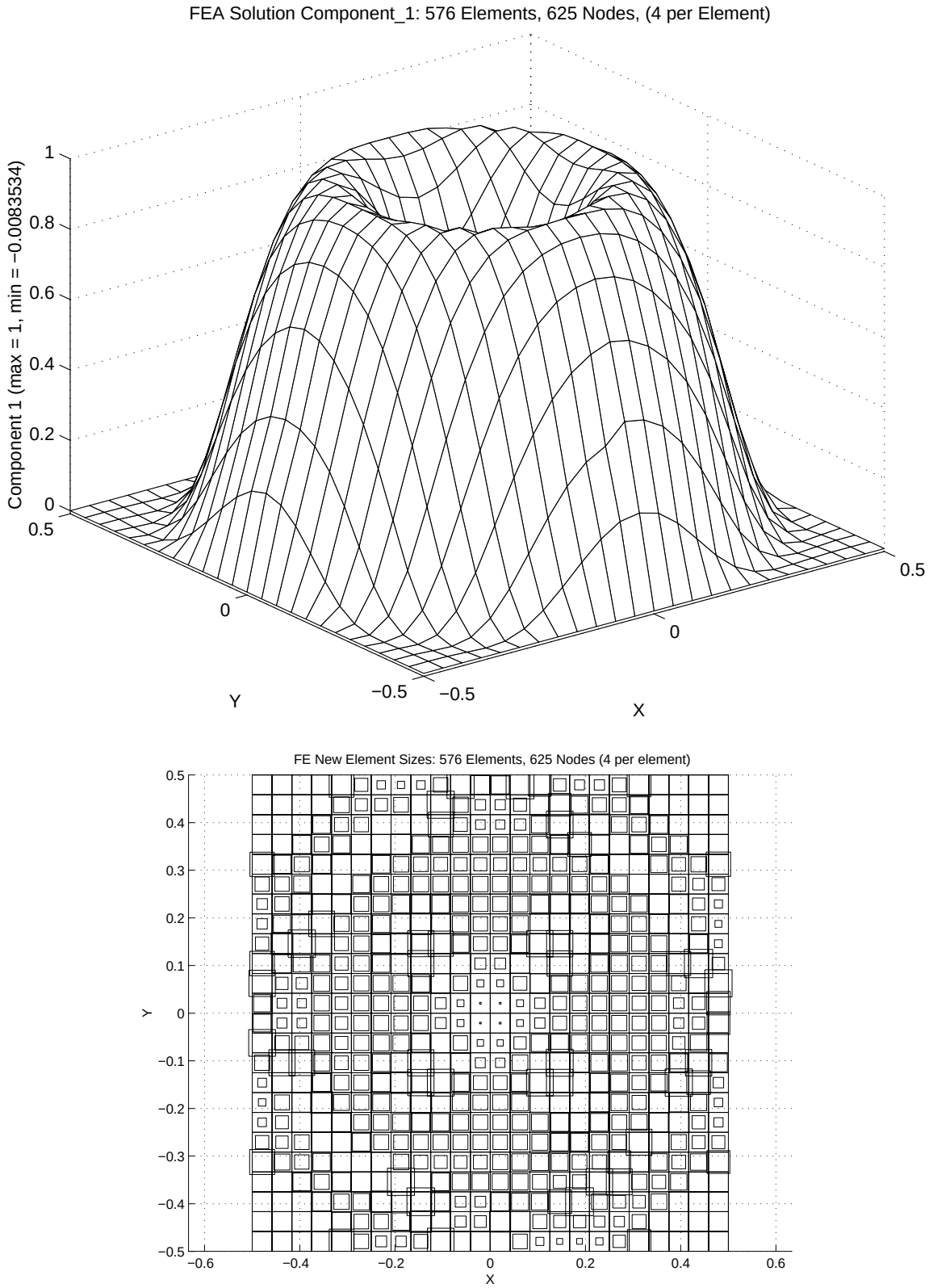


Figure 14.7.12 The Q4 result and new mesh estimate from τ_{ugn}

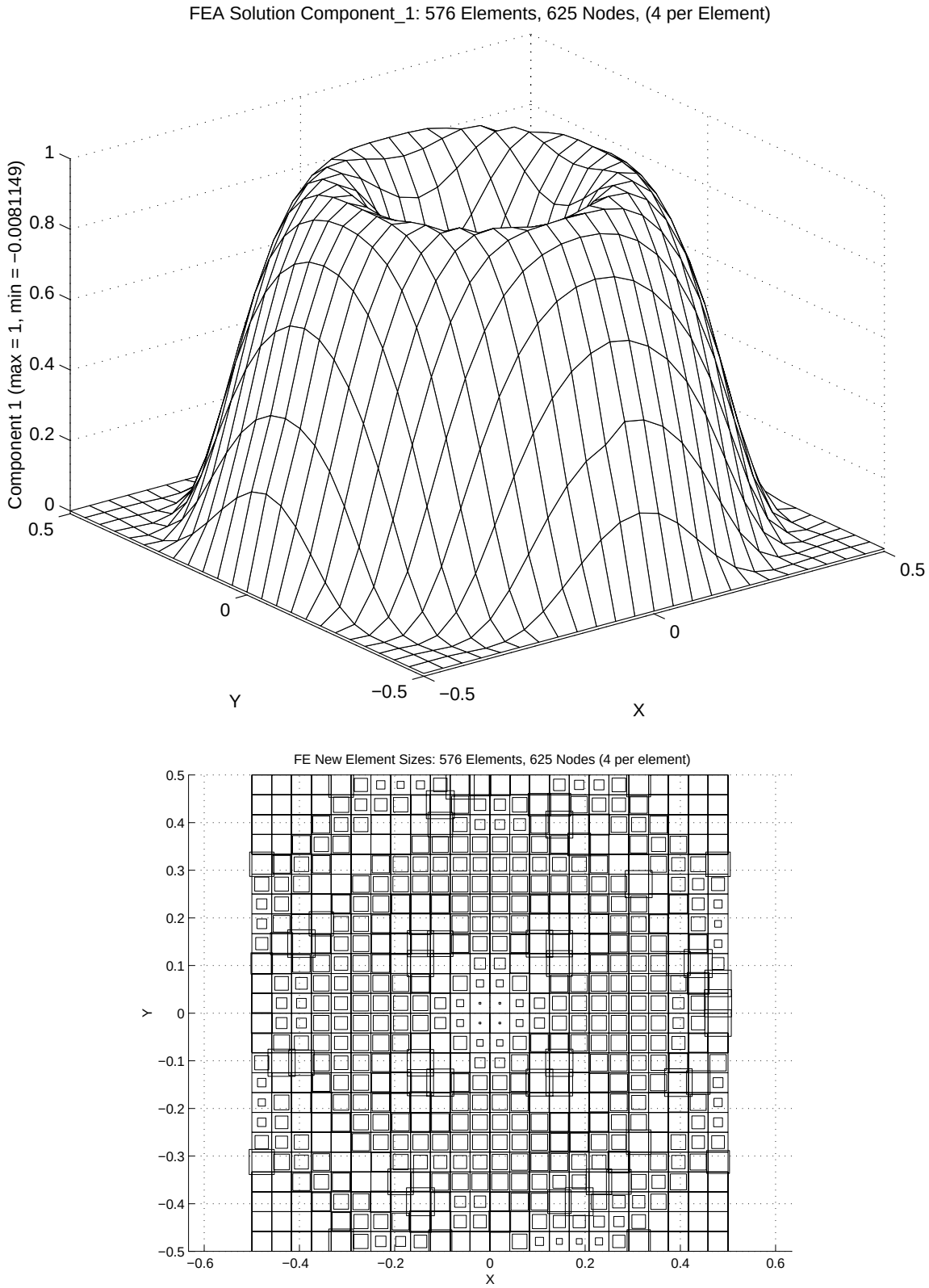
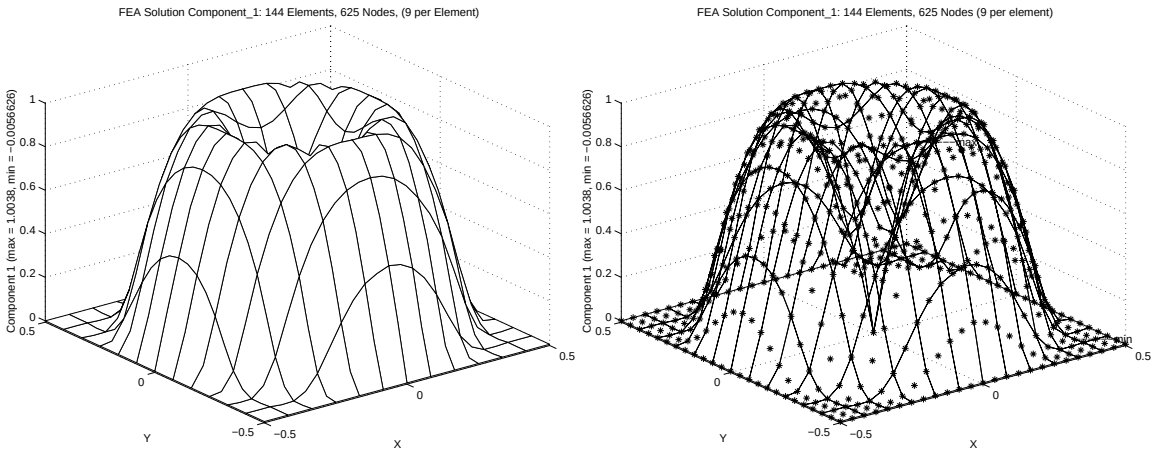


Figure 14.7.13 The Q4 result and new mesh estimate from τ_{norm}



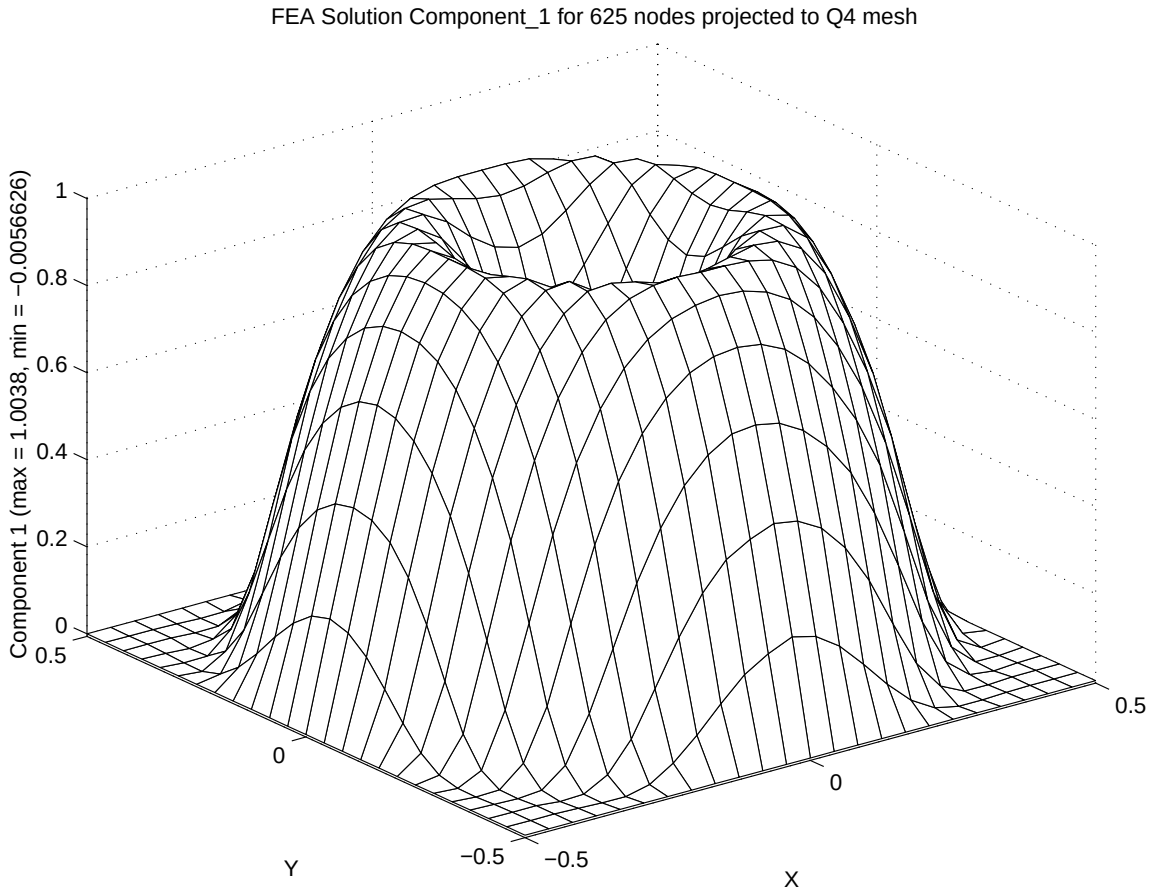
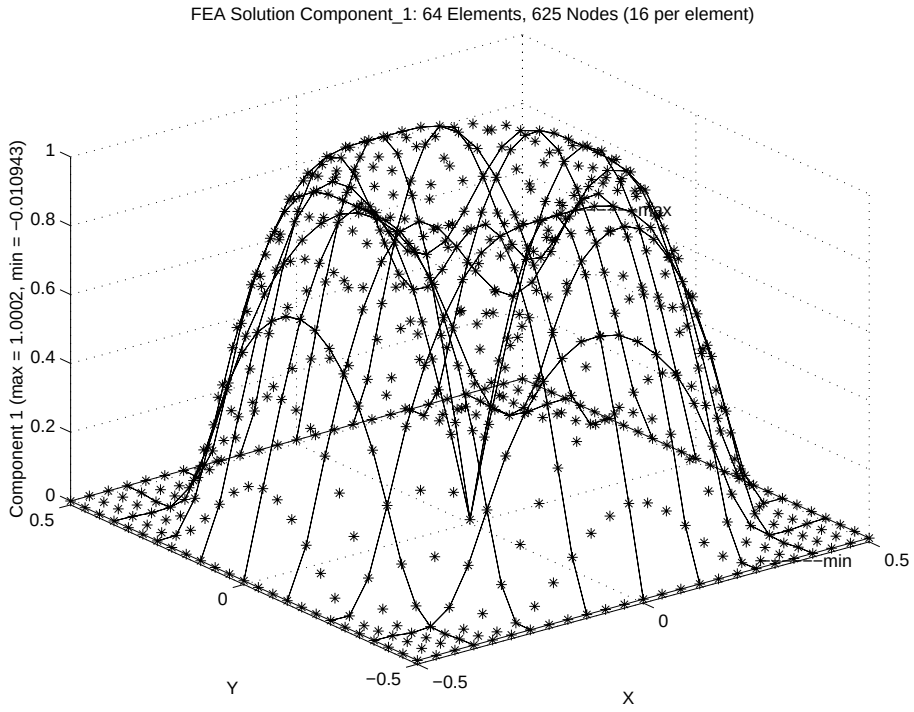


Figure 14.7.14 Quadratic Q9 results for τ_{norm}, normal, wireframe, projected



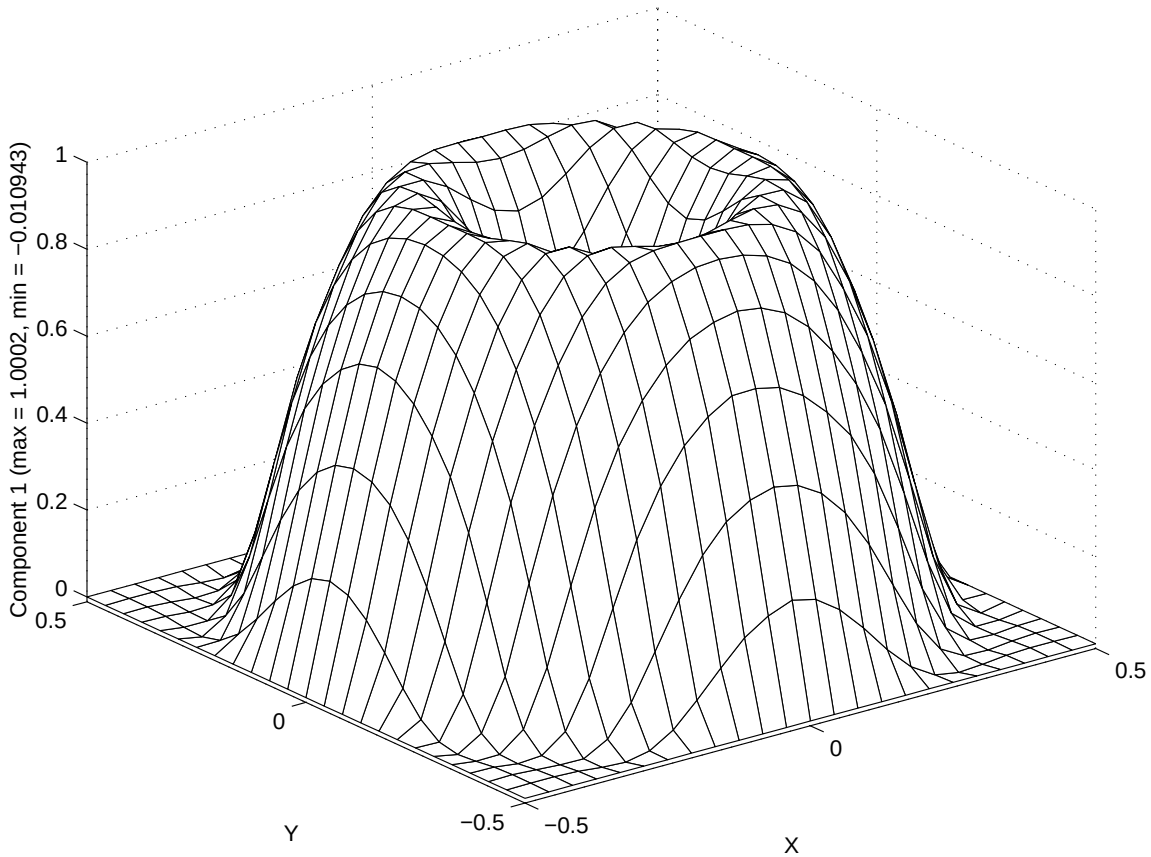


Figure 14.7.15 Cubic Q16 results for τ_{norm}, wireframe, projected

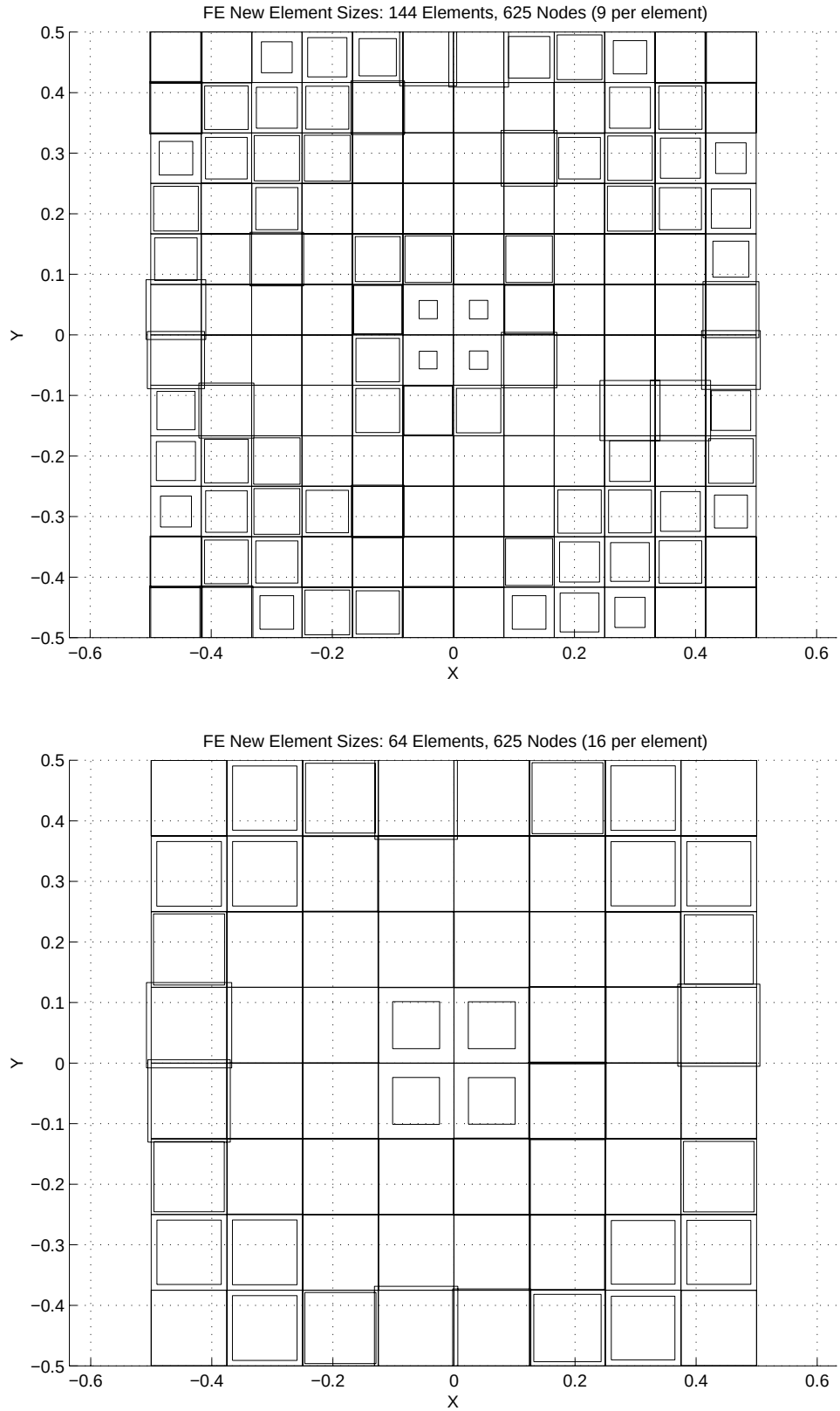


Figure 14.7.16 Initial suggested refinements for Q9, Q16 with τ_{norm}

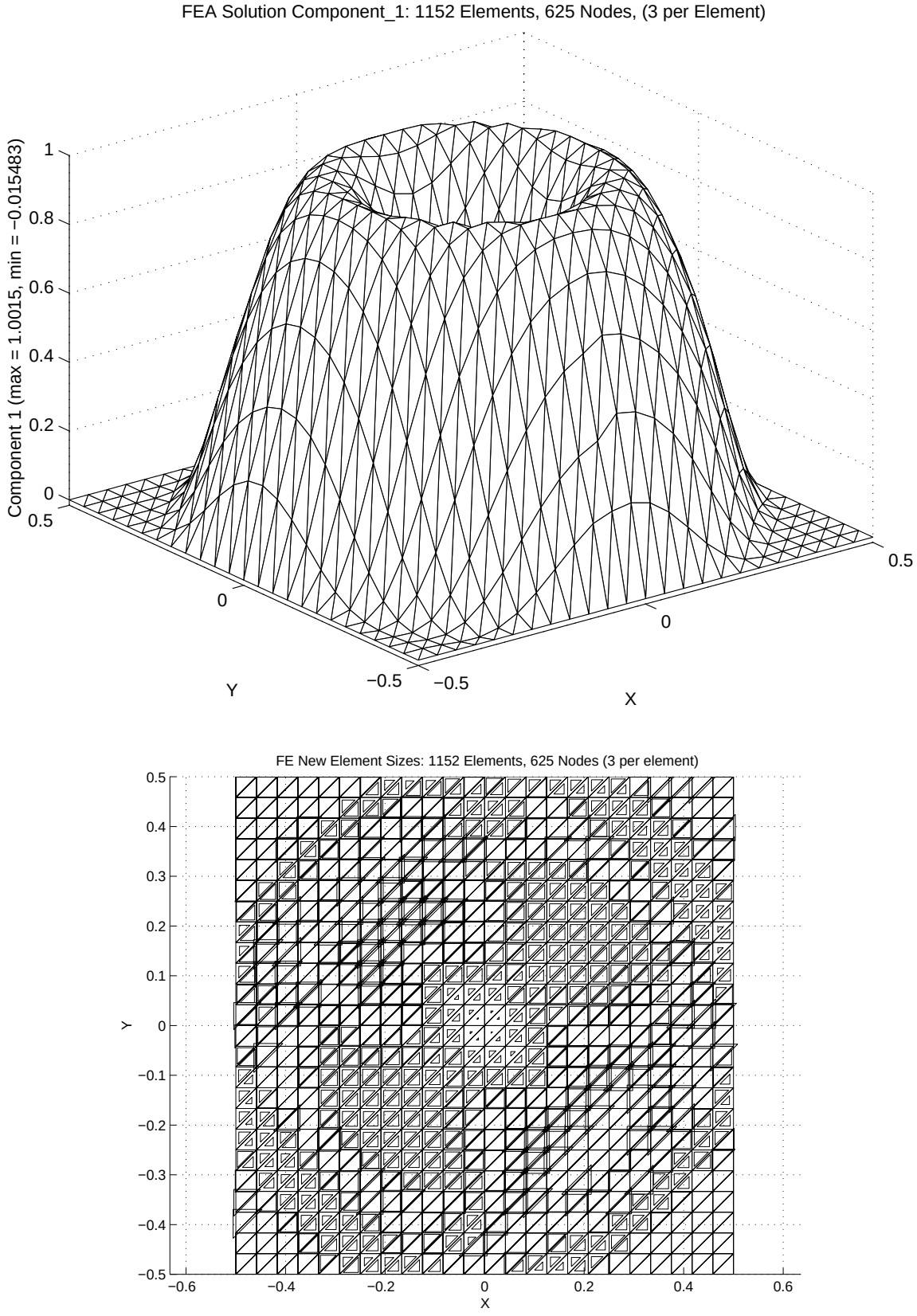


Figure 14.7.17 Linear T3 triangle results and sizes for τ_{norm}

suggested mesh refinements are given in Fig. 14.7.17 to 19, respectively.

The higher order element solutions shown here are based on the common practice of omitting the second order derivatives of the interpolation functions. All four choices for τ seem to give similar results here for a problem with no source term. One can apply the same grid spacing for meshes with Serendipity quadratic, Q8, and cubic, Q12, quadrilaterals. However, fewer nodes and equations are used since they do not have interior nodes. Considering that effect they seem to perform equally well in this test without a source term.

Since stabilization methods should also be tested on problems with source terms the final example will consider a problem with a constant source term distributed over a square, with the function having essential boundary values of zero on all four edges, and having a constant diagonal flow with a unit velocity. For very high advection rates the solution is pushed into the corner at $x = y = 1/2$ and the rapidly drops to zero through a sharp boundary layer along the lines $x = 1/2$ and $y = 1/2$. Again a uniform initial mesh is selected so that the same set of nodes can be employed in meshes that use different element types taken from the linear through cubic degree elements in list of T3 or T6 or T10 or Q4 or Q9 or Q16 elements. All of these give results that can be projected onto a common Q4 mesh to simplify visual comparisons. The initial mesh was chosen to be relatively crude and consisted of a 18×18 grid of square Q4 linear elements.

A repeatedly refined mesh of Q4 elements was employed to obtain a fine scale reference solution to which other results will be compared. Two views of that solution are given in Fig. 14.7.20. Its maximum solution value is 5.14. The four τ definitions given earlier were employed for the initial 18×18 Q4 mesh. The initial solution value contours for the reference solution, classic Galerkin, and the four τ stabilization definitions are given in Fig. 14.7.21. The peak solution values (included in the captions) are 5.14, 5.76, 4.87, 4.70, 4.99, and 5.65, respectively. When a source term is present, as here, we see that the τ_{vol} choice is like the Galerkin solution and significantly overshoots the true result by more than 10 %. The other three stabilization parameter definitions under estimate the peak value by about 5 %.

Next we will employ the same number of nodes and Q4 elements but bias the mesh toward the expected boundary layer as illustrated in Fig. 14.7.22. The boundary region of the four stabilized models, using this mesh, are given in Fig. 14.7.23 (to the same scale). The maximum solution values for τ_{norm}, τ_{ugn}, τ_{geom}, and τ_{vol} are now 4.92, 4.86, 5.07, and 5.24 respectively (compared to the much finer reference solution value of 5.14). In the initial mesh τ was a single constant over the full domain. Now since the element sizes are changing each element has a different τ value. One might expect that the stabilization terms will be highest in the boundary layer. That is not the case as can be seen from the four τ contour plots in Fig. 14.7.24. The peak τ values for each definition vary significantly. For τ_{norm}, τ_{ugn}, τ_{geom}, and τ_{vol} the peak element values are 0.0383, 0.0542, 0.0346, and 0.0018, respectively. These peak τ values occur near the largest elements in the lower left corner region. The minimum values occur at the smallest element and are 0.0058, 0.0081, 0.0011, and 0.0000, rescectively. When quadratic and cubic elements are utilized the solution improves but the distribution of τ remains about the same in shape. Their peak solution values are closer to the reference solution. The same is not true for the Serendipity Q8 and Q12 elements which yield strabge results near

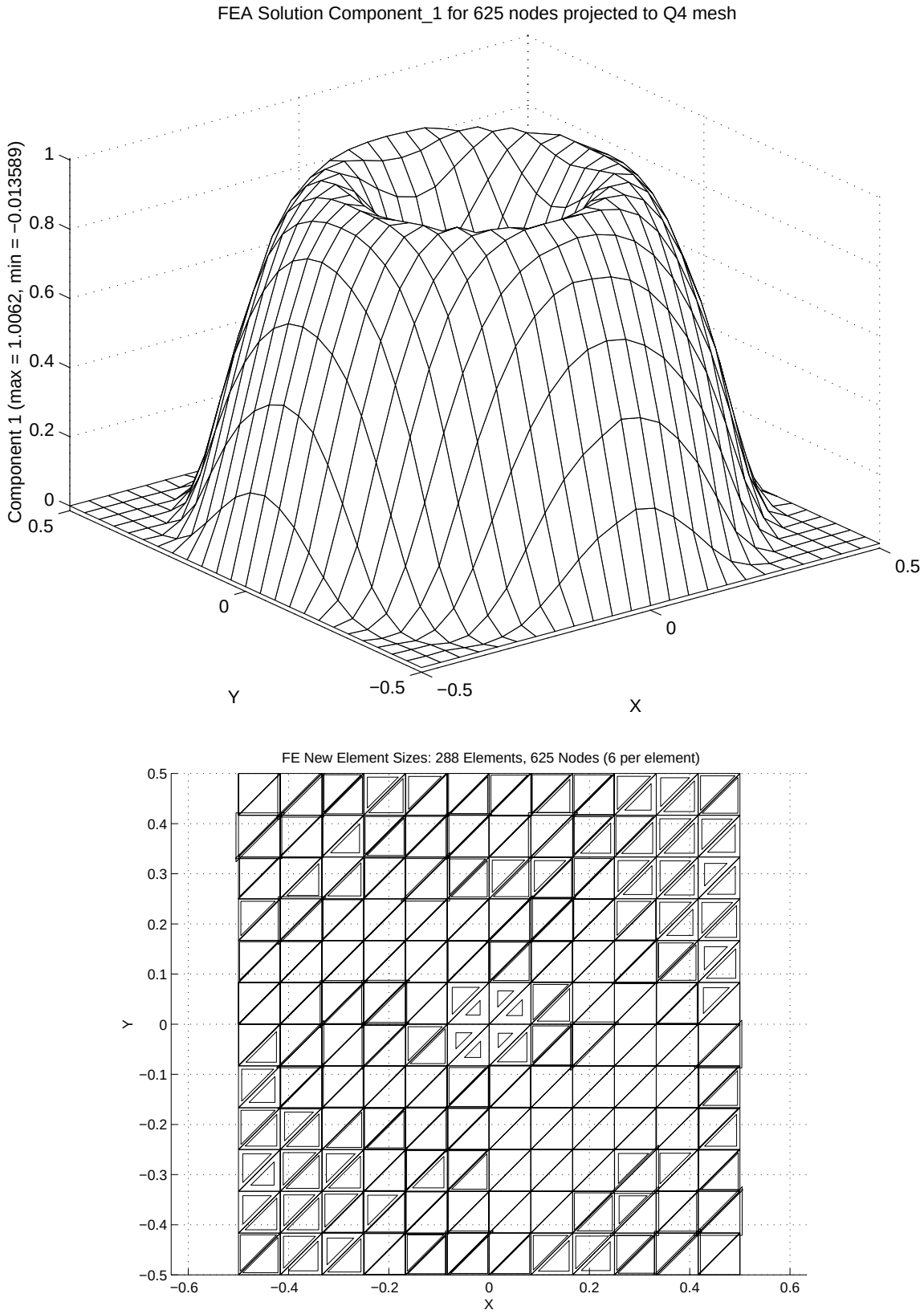


Figure 14.7.18 Projected quadratic T6 triangle results and sizes for τ_{norm}

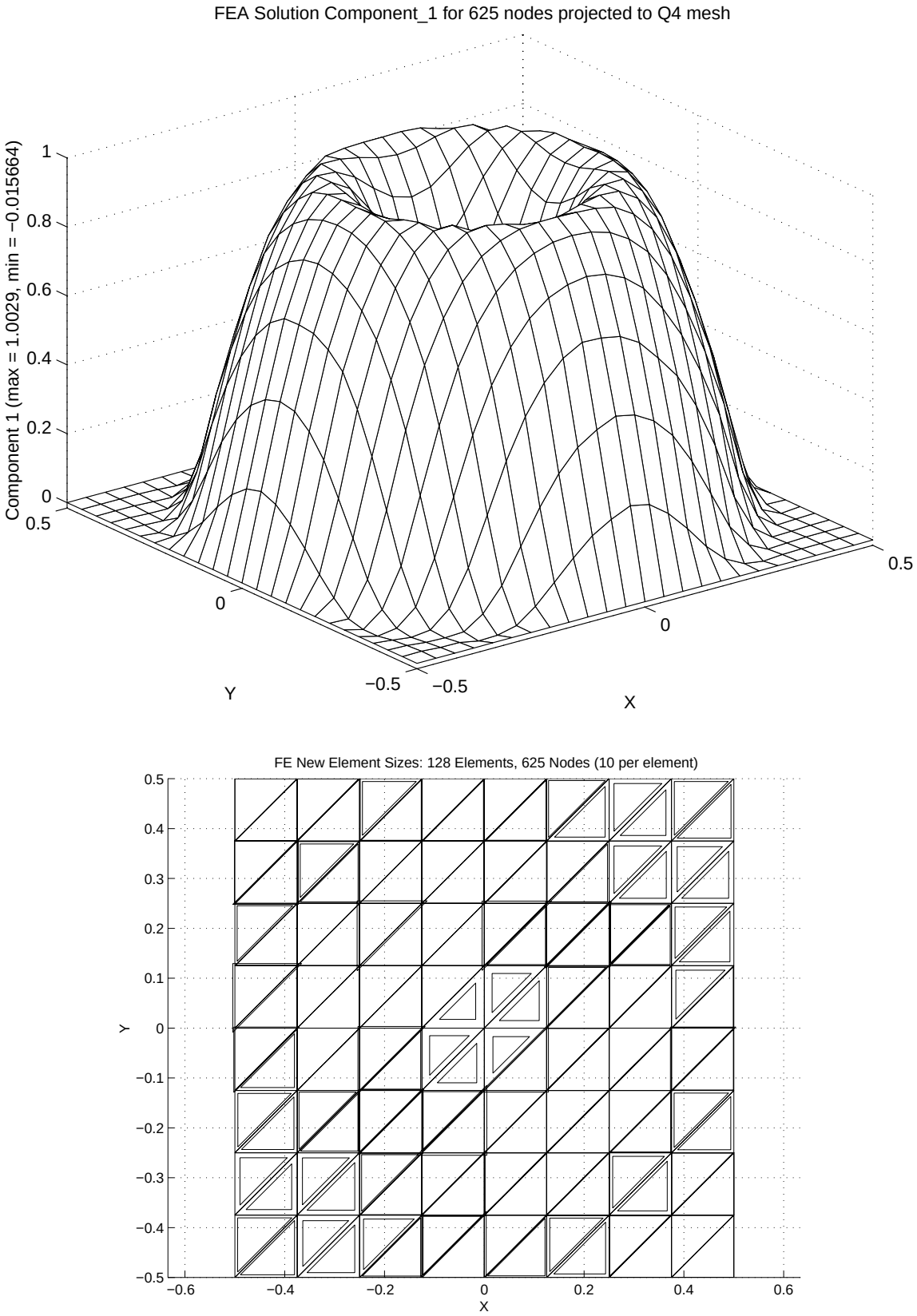
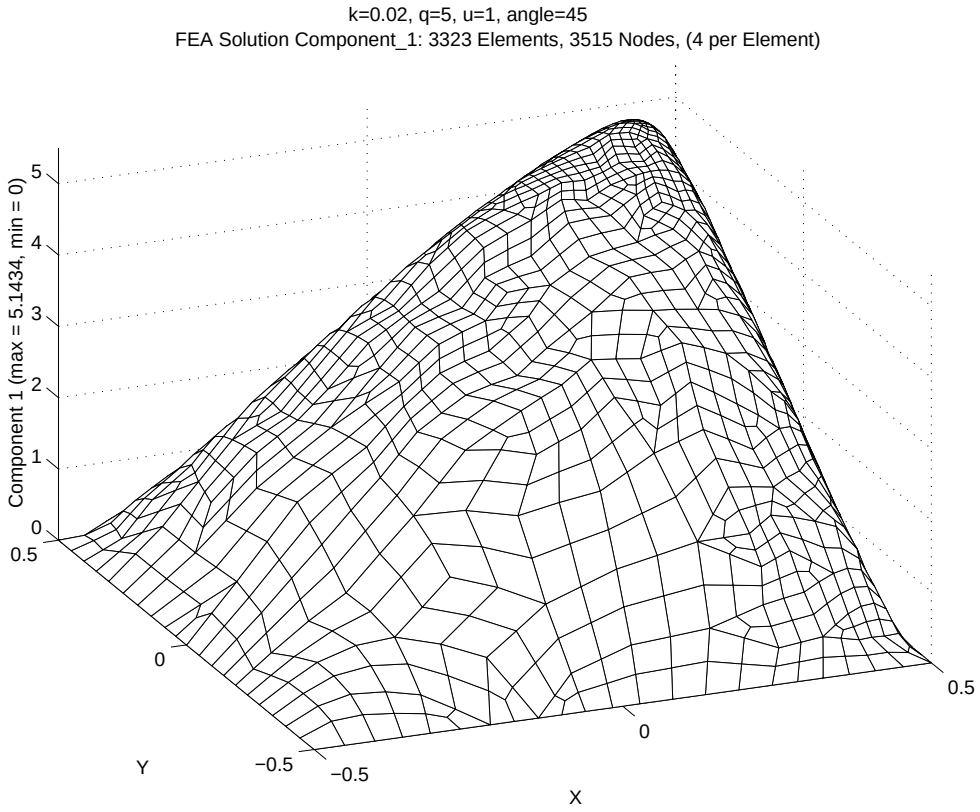


Figure 14.7.19 Projected cubic T10 triangle results and sizes for τ_{norm}



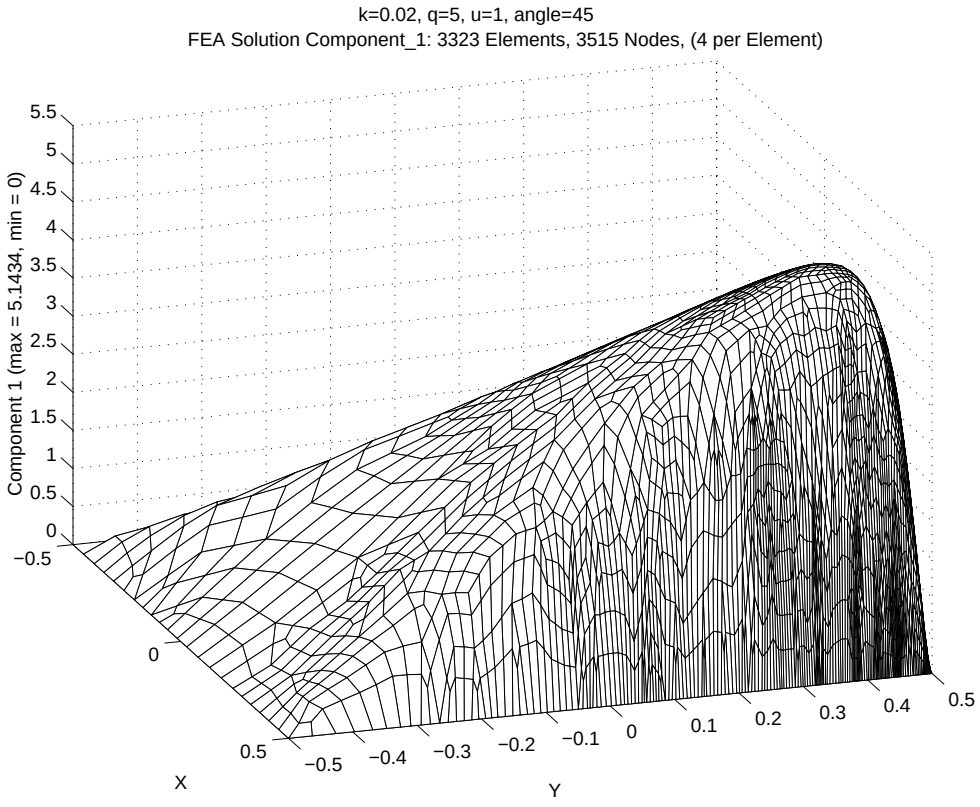


Figure 14.7.20 Reference Q4 solution for advection of a constant source

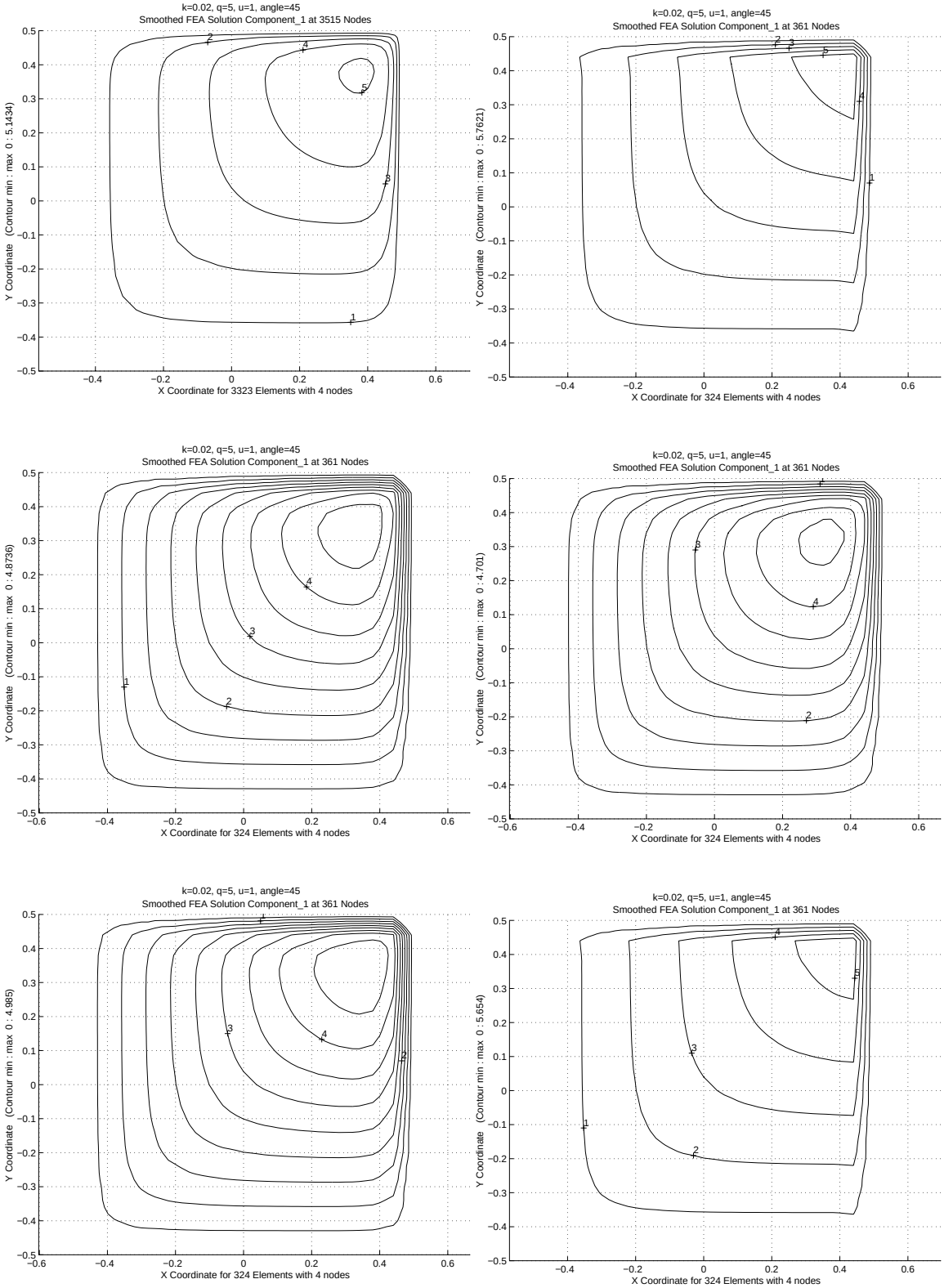


Figure 14.7.21 Contours from reference, Galerkin, τ_{norm}, τ_{ugn}, τ_{geom}, and τ_{vol}

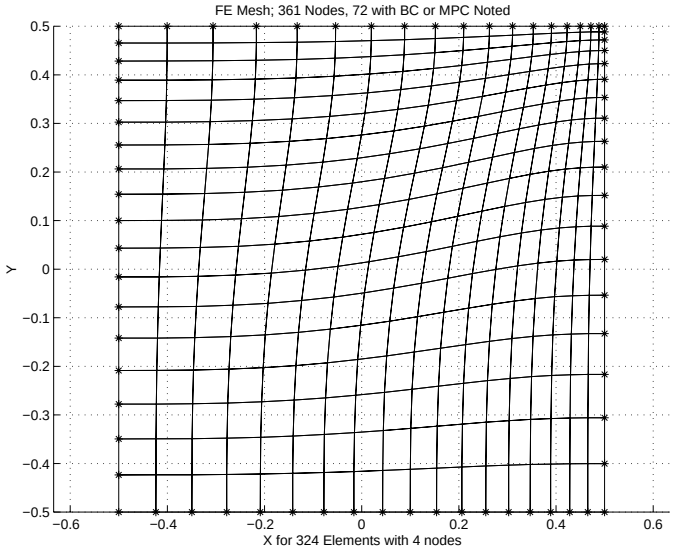


Figure 14.7.22 Varying Q4 element size for the same node count

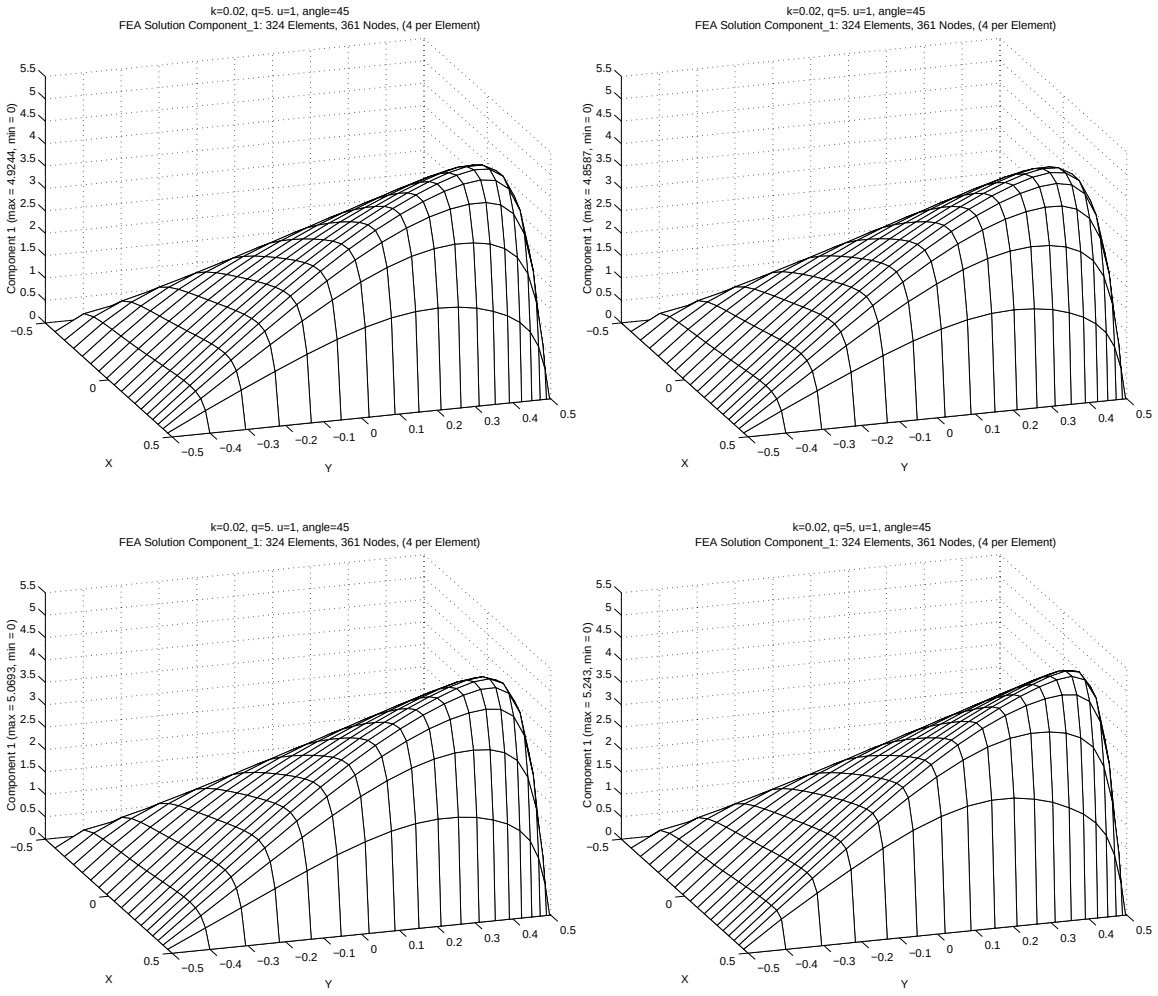


Figure 14.7.23 Boundary layer from τ_{norm}, τ_{ugn}, τ_{geom}, τ_{vol} on new Q4 mesh

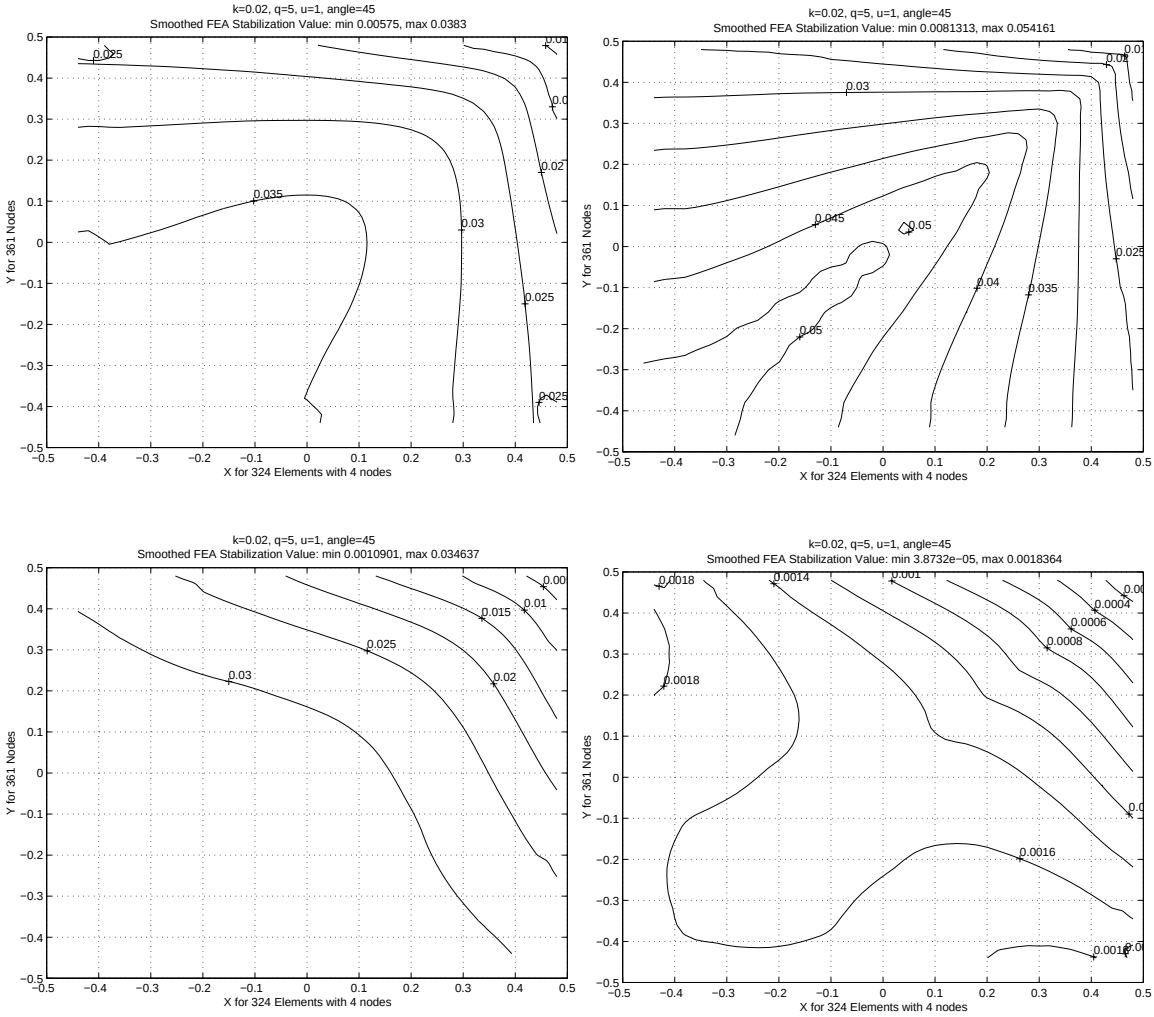


Figure 14.7.24 Distribution of the Q4 element values of τ_{norm}, τ_{ugn}, τ_{geom}, τ_{vol}

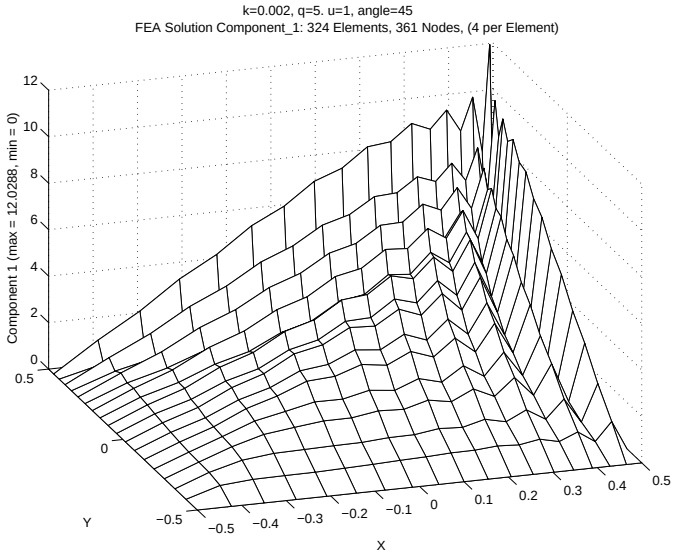


Figure 14.7.25 Unstable Galerkin Q4 solution with $k \diagup 10$

the two far corners off the main diagonal of flow.

Using mesh refinements guided by the error estimator all solutions approach the same value. Even with the above biased mesh the Galerkin solution becomes very unstable if the local Peclet number is increased as seen in Fig. 14.7.25.

The τ_{norm} and τ_{geom} can be extended to define a nodal basis τ set that define variable values at the quadrature points. They give results similar to the above approaches that hold τ constant in each element. However, the τ_{norm} approach requires iteration. That is not a major problem since iterative solvers are often used in finite element analysis, especially for Navier-Stokes solvers.

14.8 Exercises

1. A model equation with a non-uniform source and a boundary layer near $x = 1$ is $-u'' + k\,u' = Q$ when $k >> 1$ and $u(0) = 0 = u(1)$. Obtain a finite element solution when the source per unit length is defined as:

a) $Q = 3\,k\,x^2$ so that $u(x) = x^3 + A\,x^2 + B\,x + C(e^{kx} - 1)/(e^k - 1)$ where $A = 3/k$, $B = 6/k^2$, and $C = -(1 + A + B)$.

b) $Q = 3\,k\,x^2 + 2\,k\,\pi\,Cos\,(2\,\pi\,x) + 4\,\pi^2\,Sin\,(2\,\pi\,x)$ so $u(x)$ is the above expression plus $Sin\,(2\,\pi\,x)$.

Note that in both cases the total applied source is k. Use $k = 1$ and $k = 60$ to see solutions without and with a boundary layer, respectively. (For $k = 60$ these two exact solutions correspond to *exact_case* 28 and 29 in MODEL, respectively.)

2. A one-dimensional problem with boundary layer at $x = 1$ is

$$-ku''(x) + u'(x) = 1, \quad 0 \le x \le 1$$

with the boundary conditions of $u(0) = 0 = u(1)$. Obtain finite element solutions for $k = 0.1$ and $k = 0.01$ and compare them to the exact result of

$$u(x) = x - \exp[(x - 1)/k] - \exp[-1/k]/1 - \exp[-1/k].$$

14.9 References

[1] Brooks, A.N. and Hughes, T.J.R., "Streamline Upwind/Petrov-Galerkin Formulations for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations," *Comp. Meth. Appl. Mech. Engr.*, **14**, pp. 199−259 (1982).

[2] Carette, J.C., Deconinck, H., Paillere, H., and Roe, P.L., "Multidimensional Upwinding: It's Relation to Finite Elements," in *Finite Elements in Fluids*, Pineridge Press (1993).

[3] Christie, I., Griffiths, D.F., Mitchell, A.R., and Zienkiewicz, O.C., "Finite Element Methods for Second Order Equations with Significant First Derivatives," *Int. J. Num. Meth. Eng.*, **10**, pp. 1389−1396 (1976).

[4] Codina, R., Onate, E., and Cervera, M., "The Intrinsic Time for the Streamline Upwind/Petrov-Galerkin Formulation Using Quadratic Elements," *Comp. Meth. Appl. Mech. Eng.*, **94**, pp. 239−262 (1992).

[5] Donea, J., "A Review of Upwind Finite Elements," in *Finite Elements in Fluids*, Pineridge Press (1993).

[6] Franca, L.P., Frey, S.L., and Hughes, T.J.R., "Stabilized Finite Element Methods: I. Application to the Advective-Diffusion Model," *Comp. Meth. Appl. Mech. Engr.*, **59**, pp. 253−276 (1992).

[7] Gerges, H. and McCorquodale, J.A., "Modeling of Flow in Rectangular Sedimentation Tanks by an Explicit Third-Order Upwinding Technique," *Int. J. for Numerical Methods in Fluids*, **24**, pp. 537−561 (1997).

[8] Gresho, P.M., Sani, R., and Engelman, M.S., *Incompressible Flow and the Finite Element Method*, West Sussex: John Wiley (1998).

[9] Harari, I. and Hughes, T.J.R., "Stabilized Finite Element Method for Steady Advection Diffusion Equation," *Comp. Meth. Appl. Mech. Engr.*, **115**, pp. 165−191 (1994).

[10] Heinrich, J.C. and Pepper, D.W., *Intermediate Finite Element Method*, Philadelphia, PA: Taylor & Francis (1999).

[11] Huang, H.C. and Usmani, A.S., in *Finite Element Analysis for Heat Transfer*, London: Springer-Verlag (1994).

[12] Huebner, K.H. and Thornton, E.A., *Finite Element Method for Engineers*, New York: John Wiley (1982).

[13] Hughes, T.J.R., "Recent Progress in the Development and Understanding of SUPG Methods with Special Reference to the Compressible Euler and Navier-Stokes Equations," pp. 273−287 in *Finite Elements in Fluids* − Volume 7, ed. R.H. Gallagher, R. Glowinski, P.M. Gresho, J.T. Oden and O.C. Zienkiewicz, New York: John Wiley (1987).

[14] Hughes, T.J.R., Franca, L.P., and Hulbert, G.M., "A New Finite Element Formulation for Computational Fluid Dynamics, VIII: The Galerkin/Least Squares Method for Advective-Diffusion Equations," *Comp. Meth. Appl. Mech. Engr.*, **73**, pp. 173−189 (1989).

[15] Idelsohn, S.R., Nigro, N., Storti, M., and Buscaglia, G., "Petrov-Galerkin Methods for the Transient Advective Diffusion with Sharpe Gradients," *Int. Num. Meth. Eng.*, **39**, pp. 1455−1473 (1995).

[16] Kondo, N., Tosaka, N., and Nishimura, T., "Third-Order Upwind Finite Element Formulations for Incompressible Viscous Flow Problems," *Comp. Meth. Appl. Mech. Engr.*, **93**, pp. 269−187 (1991).

[17] Nassehi, V., *Practical Aspects of Finite Element Modeling of Polymer Processing*, New York: John Wiley (2002).

[18] Rice, J.G. and Schnipke, R.J., "A Monotone Streamline Upwind Finite Element Method for Convection-Dominated Flows," *Comp. Meth. Appl. Mech. Eng.*, **48**, pp. 313−327 (1985).

[19] Rice, J.G. and Schnipke, R.J., "An Equal-Order Velocity-Pressure Formulation That Does Not Exhibit Spurious Pressure Modes," *Comp. Meth. Appl. Mech. Eng.*, **58**, pp. 135−149 (1986).

[20] Shemirani, F. and Jambunathan, K., "Conservative Monotone Streamline Upwind Formulation Using Simplex Elements," *Int. J. for Numerical Methods in Fluids*, **14** , pp. 1245−1257 (1992).

[21] Tezduyar, T.E. and Hughes, T.J.R., *Finite Element Formulations for Convective Dominated Flows with Particular Emphasis on the Compressible Euler Equations*, vol. AIAA Paper 83-0125, Proc. of AIAA 21st Aerospace Sciences Meeting (1983).

[22] Tezduyar, T.E. and Park, Y.J., "Discontinuity Capturing Finite Element Formulations for Nonlinear Convection-Diffusion-Reaction Problems," *Comp. Meth. Appl. Mech. Eng.*, **59**, pp. 307−325 (1986).

[23] Tezduyar, T.E., "Stabilized Finite Element Formulations for Incompressible Flow Computations," *Advances in Applied Mechanics*, **28**, pp. 1−44 (1991).

[24] Tezduyar, T.E. and Osawa, Y., "Finite Element Stabilization Parameters Computed from Element Matrices and Vectors," *Computer Methodsin Applied Mechanics and Engineering*, **190**, pp. 411−430 (2000).

[25] Yu, C.C. and Heinrich, J.C., "Petrov−Galerkin Methods for the Time Dependent Convective Transport Equation," *Int. J. for Num. Mech. Eng.*, **23**, pp. 883−901 (1986).

[26] Zienkiewicz, O.C. and Taylor, R.L., *The Finite Element Method*, 5th Edition, London: Arnold (2000).

