IMPLEMENTATION ISSUES

FOR HIGH–ORDER FINITE ELEMENT METHODS *

Babak Bagheri
Computer Science Department, Penn State University

L. Ridgway Scott
Department of Mathematics, University of Houston

Shangyou Zhang
Department of Mathematics, University of Delaware

Presented at the *Second International Conference on Spectral and High Order Methods*, Montpelier, France, June 1992.

High-order methods provide efficient ways to model complex physical phenomena such as incompressible fluid flows. However, computer codes for high-order finite elements are significantly more challenging to develop. We present some results obtained with two codes, `fec` and `nmg`, that use high-order finite-element methods, and we discuss some of the design issues that affected the development of the codes, focusing on those issues related to high-order finite elements. We illustrate the use of `fec` applied to the Navier-Stokes equations in two dimensions and give some results obtained with `nmg`, which implements non-nested multilevel iterative techniques in three dimensions for high-order elements.

`fec` was written in the language, C++, and uses advanced ideas from computer science to simplify implementation and to minimize the cost of maintaining and extending the code (Scott & Bagheri, 1990). The extensibility of C++ makes the use of `fec` essentially like using a separate language for finite elements, in that objects like interpolants, bilinear forms, and norms are explicitly available.

`nmg` is an implementation (in Fortran) of the algorithms discussed in Scott & Zhang (1992). It is an interactive code that allows one to experiment with different multilevel iterative methods for solving the linear system associated with finite element methods for elliptic problems in three dimensions.

* This work was supported in part by the National Science Foundation through award number DMS-9105437.

The use of higher-order elements affects implementation strategies for computer codes in several ways. Key issues include the generation of basis functions and quadrature rules and the evaluation of bilinear forms. Whereas different approaches have essentially the same order of complexity for low-order elements, they become strongly differentiated as the degree of approximation increases. We present some analysis of the different algorithms that can be used to evaluate bilinear forms that was influential in the development of `fec` and `nmg`.

One key to the success of the finite element method as developed in engineering practice was the systematic way that computer codes for it could be implemented. One important step in this process is the *assembly* of integrated differential forms by summing its constituent parts over each *element*, which are computed separately. There are two issues to consider. First, the element-level forms must be computed efficiently. Second, one must decide, say for a bilinear form, whether to compute and store a global stiffness matrix or to compute global forms just by summing element-level forms at each occurrence.

We study two different algorithms available to evaluate element-level multilinear forms arising in the approximation of the Navier-Stokes equations. This system can be viewed as a prototype as it involves both bilinear and trilinear forms. (In general, k-linear forms are considered for arbitrary k.) We show that it can be more efficient, both in time and storage, *not* to use element matrices as the degree of approximating functions increases.

In evaluating bilinear forms, one has the choice of forming the full stiffness matrix, or not. Once the stiffness matrix is formed, evaluating bilinear forms becomes more efficient than using element-level forms. We study the cost of assembly of the stiffness matrix versus direct evaluation of bilinear forms in two and three dimensions. We show that it can be more efficient *not* to assemble the full stiffness matrix for high-order methods in some cases.

We consider the computational issues with regard to both Euclidean and isoparametric elements. We show that isoparametric elements may be used with higher-order elements without increasing the order of computational complexity.

1. Evaluation of Local Bilinear Forms

The *assembly* of integrated differential forms by summing its constituent parts over each *element*, which are computed separately is facilitated through the use of a numbering scheme called the *local-to-global* index. This index, $\iota(e, \lambda)$, relates the local (or element) node number, $\lambda \in \mathcal{L}$, on a particular element, indexed by e, to its position in the global data structure.

We may write the interpolant of a continuous function for the space of all piecewise polynomial (of some degree) functions (no boundary conditions imposed) via

$$f_I := \sum_e \sum_{\lambda \in \mathcal{L}} f(x_{\iota(e,\lambda)}) \phi_\lambda^e \tag{1.1}$$

where $\{\phi_\lambda^e : \lambda \in \mathcal{L}\}$ denotes the set of basis functions for polynomials of the given degree on T_e. We can relate all of the "element" basis functions ϕ_λ^e to a fixed set of basis functions on a "reference" element, $\mathcal{T}$, via an affine mapping, $\xi \to J\xi + x_e$, of $\mathcal{T}$ to T_e:

$$\phi_\lambda^e(x) = \phi_\lambda \left(J^{-1}(x - x_e) \right).$$

(By definition, the element basis functions, ϕ_λ^e, are extended by zero outside T_e.) The inverse mapping, $x \to \xi = J^{-1}(x - x_e)$ has as its Jacobian

$$J_{mj}^{-1} = \frac{\partial \xi_m}{\partial x_j}$$

and this is the quantity which appears in the evaluation of the bilinear forms. Of course, $\det J = 1/\det J^{-1}$.

All of the multilinear forms arising in the variational formulation of differential equations can be easily evaluated (assembled) using this representation as well. We consider the Navier-Stokes equations (in d dimensions) as they involve both bilinear and trilinear forms. For example,

$$a(\mathbf{v}, \mathbf{w}) = \sum_e a_e(\mathbf{v}, \mathbf{w})$$

where the "element" bilinear form is defined (and evaluated) via

$$
\begin{aligned}
a_e(\mathbf{v},\mathbf{w}) &:= \int_{T_e} \nabla\mathbf{v}(x) : \nabla\mathbf{w}(x)\,dx \\
&= \int_{T_e} \sum_{k=1}^{d} \nabla v_k(x)\cdot\nabla w_k(x)\,dx \\
&= \int_{\mathcal{T}} \sum_{j,k=1}^{d} \frac{\partial}{\partial x_j} v_k(J\xi + x_e) \frac{\partial}{\partial x_j} w_k(J\xi + x_e) \det(J)\,d\xi \\
&= \int_{\mathcal{T}} \sum_{j,k,m,m'=1}^{d} \frac{\partial \xi_m}{\partial x_j}\frac{\partial}{\partial \xi_m}\left(\sum_{\lambda\in\mathcal{L}} v_k^{\iota(e,\lambda)}\phi_\lambda(\xi)\right) \times \\
&\quad \frac{\partial \xi_{m'}}{\partial x_j}\frac{\partial}{\partial \xi_{m'}}\left(\sum_{\mu\in\mathcal{L}} w_k^{\iota(e,\mu)}\phi_\mu(\xi)\right)\det(J)\,d\xi \\
&= \sum_{k=1}^{d} \begin{pmatrix} v_k^{\iota(e,1)} \\ \cdot \\ \cdot \\ v_k^{\iota(e,|\mathcal{L}|)} \end{pmatrix}^{t} \mathbf{K}^e \begin{pmatrix} w_k^{\iota(e,1)} \\ \cdot \\ \cdot \\ w_k^{\iota(e,|\mathcal{L}|)} \end{pmatrix}.
\end{aligned}
$$

Here, the *element stiffness matrix*, $\mathbf{K}^e$, is given by

$$
\begin{aligned}
K^e_{\lambda,\mu} &:= \sum_{j,m,m'=1}^{d} \frac{\partial \xi_m}{\partial x_j}\frac{\partial \xi_{m'}}{\partial x_j}\det(J)\int_{\mathcal{T}} \frac{\partial}{\partial \xi_m}\phi_\lambda(\xi)\frac{\partial}{\partial \xi_{m'}}\phi_\mu(\xi)\,d\xi \\
&= \sum_{j,m,m'=1}^{d} \frac{\partial \xi_m}{\partial x_j}\frac{\partial \xi_{m'}}{\partial x_j}\det(J) K_{\lambda,\mu,m,m'}
\end{aligned}
$$

for $\lambda,\mu\in\mathcal{L}$. Note that we have identified the space of piecewise polynomials functions, $\mathbf{v}$, with the vector space of values, $(\mathbf{v}^i)$, at the nodes.

The nonlinear term in the Navier–Stokes term can also be treated similarly, using "element" trilinear forms, as follows.

$$
\begin{aligned}
c_e(\mathbf{u},\mathbf{v},\mathbf{w}) &:= \int_{T_e} \mathbf{u}\cdot\nabla\mathbf{v}(x)\cdot\mathbf{w}(x)\,dx \\
&= \int_{T_e} \sum_{j,k=1}^{d} u_j(x)\frac{\partial}{\partial x_j} v_k(x) w_k(x)\,dx \\
&= \int_{\mathcal{T}} \sum_{j,k=1}^{d} u_j(J\xi+x_e)\frac{\partial}{\partial x_j} v_k(J\xi+x_e) w_k(J\xi+x_e)\det(J)\,d\xi \\
&= \int_{\mathcal{T}} \sum_{j,k,m=1}^{d} \left(\sum_{\lambda\in\mathcal{L}} u_j^{\iota(e,\lambda)}\phi_\lambda(\xi)\right)\times \\
&\quad \frac{\partial\xi_m}{\partial x_j}\left(\sum_{\mu\in\mathcal{L}} v_k^{\iota(e,\mu)}\frac{\partial}{\partial\xi_m}\phi_\mu(\xi)\right)\left(\sum_{\rho\in\mathcal{L}} w_k^{\iota(e,\rho)}\phi_\rho(\xi)\right)\det(J)\,d\xi \\
&= \sum_{j,k,m=1}^{d}\sum_{\lambda,\mu,\rho\in\mathcal{L}} u_j^{\iota(e,\lambda)}\frac{\partial\xi_m}{\partial x_j} v_k^{\iota(e,\mu)} w_k^{\iota(e,\rho)}\det(J)\times \\
&\quad \int_{\mathcal{T}} \phi_\lambda(\xi)\frac{\partial}{\partial\xi_m}\phi_\mu(\xi)\phi_\rho(\xi)\,d\xi \\
&= \sum_{j,k=1}^{d}\sum_{\lambda,\mu,\rho\in\mathcal{L}} u_j^{\iota(e,\lambda)} v_k^{\iota(e,\mu)} w_k^{\iota(e,\rho)}\times \\
&\quad \sum_{m=1}^{d}\frac{\partial\xi_m}{\partial x_j}\det(J)\int_{\mathcal{T}} \phi_\lambda(\xi)\frac{\partial}{\partial\xi_m}\phi_\mu(\xi)\phi_\rho(\xi)\,d\xi \\
&= \sum_{j,k=1}^{d}\sum_{\lambda,\mu,\rho\in\mathcal{L}} u_j^{\iota(e,\lambda)} v_k^{\iota(e,\mu)} w_k^{\iota(e,\rho)}\sum_{m=1}^{d}\frac{\partial\xi_m}{\partial x_j}\det(J)N_{\lambda,\mu,\rho,m} \\
&= \sum_{j,k=1}^{d}\sum_{\lambda,\mu,\rho\in\mathcal{L}} u_j^{\iota(e,\lambda)} v_k^{\iota(e,\mu)} w_k^{\iota(e,\rho)} N^e_{\lambda,\mu,\rho,j}
\end{aligned}
$$

where

$$N_{\lambda,\mu,\rho,m} := \int_{\mathcal{T}} \phi_\lambda(\xi)\frac{\partial}{\partial\xi_m}\phi_\mu(\xi)\phi_\rho(\xi)\,d\xi$$

and

$$N^e_{\lambda,\mu,\rho,j} := \sum_{m=1}^{d}\frac{\partial\xi_m}{\partial x_j}\det(J)N_{\lambda,\mu,\rho,m}.$$

In the special case $\mathbf{u} = \mathbf{v}$, this expression simplifies:

$$c_e(\mathbf{u},\mathbf{u},\mathbf{w}) = \sum_{k=1}^{d}\sum_{\rho\in\mathcal{L}}\left(\sum_{j=1}^{d}\sum_{\lambda,\mu\in\mathcal{L}} u_j^{\iota(e,\lambda)} u_k^{\iota(e,\mu)} \sum_{m=1}^{d}\frac{\partial\xi_m}{\partial x_j}\det(J) N_{\lambda,\mu,\rho,m}\right) w_k^{\iota(e,\rho)}$$
$$= \sum_{k=1}^{d}\sum_{\rho\in\mathcal{L}} U_k^{\iota(e,\rho)} w_k^{\iota(e,\rho)}.$$

Note that U has the same structure as an "interpolant."

Note that both $N_{\lambda,\mu,\rho,(\cdot)}$ and $N^e_{\lambda,\mu,\rho,(\cdot)}$ can be thought of as d–vectors. Moreover

$$N^e_{\lambda,\mu,\rho,(\cdot)} = \det(J) N_{\lambda,\mu,\rho,(\cdot)} J^{-1}.$$

Also note that $N_{\lambda,\mu,\rho,(\cdot)} = N_{\rho,\mu,\lambda,(\cdot)}$, so that considerable storage reduction could be made if desired.

There is another way to evaluate multilinear forms that can be more efficient both in time and storage than the above approach using element matrices. Let us return to the example involving $a(\cdot,\cdot)$. We have

$$a_e(\mathbf{v},\mathbf{w}) = \int_{\mathcal{T}} \sum_{j,k=1}^{d}\left(\left(J^{-1}\right)^t \sum_{\lambda\in\mathcal{L}} v_k^{\iota(e,\lambda)}\nabla\phi_\lambda(\xi)\right)_j \times$$
$$\left(\left(J^{-1}\right)^t \sum_{\lambda\in\mathcal{L}} w_k^{\iota(e,\lambda)}\nabla\phi_\lambda(\xi)\right)_j \det(J)\, d\xi$$

Assuming we use a quadrature rule to evaluate the integral:

$$\int_{\mathcal{T}} f(\xi)\, d\xi := \sum_{\xi\in\Xi}\omega_\xi f(\xi)$$

we can convert the above as follows:

$$a_e(\mathbf{v},\mathbf{w}) = \sum_{\xi\in\Xi}\omega_\xi \sum_{j,k=1}^{d}\left(\left(J^{-1}\right)^t \sum_{\lambda\in\mathcal{L}} v_k^{\iota(e,\lambda)}\nabla\phi_\lambda(\xi)\right)_j \times$$
$$\left(\left(J^{-1}\right)^t \sum_{\lambda\in\mathcal{L}} w_k^{\iota(e,\lambda)}\nabla\phi_\lambda(\xi)\right)_j \det(J)\,.$$

The work estimate for the latter approach is of the order $|\Xi| \times |\mathcal{L}|$ whereas the previous approach is of the order $|\mathcal{L}|^2$. Depending on the number of quadrature points versus the number of basis functions, this approach may be more or less efficient. However, when we extend it to apply to the trilinear form $c(\cdot,\cdot)$ it becomes clearly superior. In this case we have

$$c_e(\mathbf{u},\mathbf{v},\mathbf{w}) = \sum_{\xi\in\Xi}\omega_\xi \sum_{j,k=1}^{2}\left(\sum_{\lambda\in\mathcal{L}} u_j^{\iota(e,\lambda)}\phi_\lambda(\xi)\right)\times$$
$$\left(\left(J^{-1}\right)^t \sum_{\lambda\in\mathcal{L}} v_k^{\iota(e,\lambda)}\nabla\phi_\lambda(\xi)\right)_j \left(\sum_{\lambda\in\mathcal{L}} w_k^{\iota(e,\lambda)}\phi_\lambda(\xi)\right)\det(J)\,.$$

The work estimate for the latter approach remains of the order $|\Xi|\times|\mathcal{L}|$ whereas the previous approach is of the order $|\mathcal{L}|^3$. In general, the latter approach applied to a κ-linear form will have a work estimate of the form $\kappa\times|\Xi|\times|\mathcal{L}|$ whereas the previous approach would be of the order $|\mathcal{L}|^\kappa$.

THEOREM 1.1. The local κ-linear forms can be computed in an amount of work proportional to the product of κ, the number of quadrature points, and the number of degrees of freedom of the local basis functions.

2. Evaluation of Bilinear Forms for Isoparametrics Elements

When using high-order elements for problems with curved boundaries, it is essential to include some way of approximating the boundary conditions accurately. One of the most effective in engineering practice is to use isoparametrics elements. In this approach, the element basis functions ϕ_λ^e are related to the reference basis functions via a polynomial mapping, $\xi\to F(\xi)$, of $\mathcal{T}$ to T_e:

$$\phi_\lambda^e(x) = \phi_\lambda\left(F^{-1}(x)\right).$$

The use of element matrices becomes quite complicated with isoparametrics elements.

Let us return to the example involving $a(\cdot,\cdot)$. We have

$$
\begin{aligned}
a_e(\mathbf{v},\mathbf{w}) &= \int_{\mathcal{T}} \sum_{j,k=1}^{d} \left(J_F^{-1}(\xi)^t \sum_{\lambda\in\mathcal{L}} v_k^{\iota(e,\lambda)} \nabla\phi_\lambda(\xi) \right)_j \times \\
&\qquad \left(J_F^{-1}(\xi)^t \sum_{\lambda\in\mathcal{L}} w_k^{\iota(e,\lambda)} \nabla\phi_\lambda(\xi) \right)_j \det(J_F(\xi))\, d\xi \\
&= \sum_{\xi\in\Xi} \omega_\xi \sum_{j,k=1}^{d} \left(J_F^{-1}(\xi)^t \sum_{\lambda\in\mathcal{L}} v_k^{\iota(e,\lambda)} \nabla\phi_\lambda(\xi) \right)_j \times \\
&\qquad \left(J_F^{-1}(\xi)^t \sum_{\lambda\in\mathcal{L}} w_k^{\iota(e,\lambda)} \nabla\phi_\lambda(\xi) \right)_j \det(J_F(\xi)) \,.
\end{aligned}
$$

Writing this in terms of element matrices is quite similar to using trilinear forms, i.e., $a_e(u,v) = a_e(F;u,v)$. Typically, we would write

$$
F_k(\xi) = \sum_{\rho\in\mathcal{L}} F_k^{\iota(e,\rho)} \phi_\rho(\xi)
$$

so that

$$
J_F(\xi)_{k,m} = \frac{\partial}{\partial\xi_m} F_k(\xi) = \sum_{\rho\in\mathcal{L}} F_k^{\iota(e,\rho)} \frac{\partial}{\partial\xi_m} \phi_\rho(\xi).
$$

The above expression for a_e is then a trilinear expression in v_k, w_k and F_k. The cost of evaluating this in terms of element matrices would thus be on the order of $|\mathcal{L}|^3$.

The other way to evaluate multilinear forms can be more efficient than using element matrices. The cost of evaluating $J_F(\xi)$ as above is only on the order of $|\Xi|\cdot|\mathcal{L}|$. The amount of work involved in the inversion and transpose of J is independent of $|\Xi|$ and $|\mathcal{L}|$. Therefore, the work estimate for the latter approach remains of the order $|\Xi|\cdot|\mathcal{L}|$.

We can apply this approach to the trilinear form $c(\cdot,\cdot,\cdot)$ as well:

$$
\begin{aligned}
c_e(\mathbf{u},\mathbf{v},\mathbf{w}) = \sum_{\xi\in\Xi} \omega_\xi \sum_{j,k=1}^{d} \left(\sum_{\lambda\in\mathcal{L}} u_j^{\iota(e,\lambda)} \phi_\lambda(\xi) \right) \times \\
\left(J_F^{-1}(\xi)^t \sum_{\lambda\in\mathcal{L}} v_k^{\iota(e,\lambda)} \nabla\phi_\lambda(\xi) \right)_j \left(\sum_{\lambda\in\mathcal{L}} w_k^{\iota(e,\lambda)} \phi_\lambda(\xi) \right) \det(J_F(\xi)) \,.
\end{aligned}
$$

This has the same order $|\Xi|\cdot|\mathcal{L}|$ work estimate. Thus the asymptotic work estimate does not change with the introduction of isoparametric elements.

THEOREM 2.1. The local κ-linear forms for isoparameteric elements can be computed in an amount of work proportional to the product of κ, the number of quadrature points, and the number of degrees of freedom of the local basis functions.

3. Evaluation of Bilinear Forms Using a Global Matrix

The evaluation of a global bilinear form such as $a(\mathbf{v},\mathbf{w})$ can be done via a global stiffness matrix, A, as

$$a(\mathbf{v},\mathbf{w}) = \sum_{i,j,k,\ell} A_{i,j}^{k,\ell} v_k^i w_\ell^j$$

where $A_{i,j}^{k,\ell}$ is defined in the obvious way. Here $k = 1,\ldots,d$ and i and j range over the set, $\mathcal{I}$, of global indices $\iota(e,\lambda)$. Thus it is a $|\mathcal{I}| \times |\mathcal{I}|$ block matrix, with $d \times d$ blocks. (Since there are no nonzero cross terms, $v_k^i w_\ell^j$, for $k \neq \ell$ these blocks are all diagonal, in fact, a constant times the identity.)

The previous approach involved an amount of work proportional to $|\mathcal{L}|^2 \times |\mathcal{E}|$ where $\mathcal{E}$ is the set of all elements, independent of whether a local element matrix is used or not.

Due to the sparseness of A, the cost of using a global matrix is more difficult to estimate. It can be done in $|A|$ operations, where $|A|$ denotes the number of nonzeros. However, it is easy to see that this is also proportional to $|\mathcal{L}|^2 \times |\mathcal{E}|$ since the entries corresponding to the interior nodes in each element are of this order. Therefore, both approaches offer asymptotically the same efficiency for higher-order methods.

The two dimensional case

In the two dimensional case, we can compute the work estimates for the global-matrix approach to the element-matrix approach with regard to the constants appearing in front of $|\mathcal{L}|^2 \times |\mathcal{E}|$. The work to evaluate $a(\mathbf{v},\mathbf{w})$ using element matrices is precisely $d \times |\mathcal{L}|^2 \times |\mathcal{E}|$. If we are dealing with polynomials of degree k, then $|\mathcal{L}| = \frac{1}{2}(k+1)(k+2)$. Thus the total work is precisely

$$d \times \frac{1}{4}(k+2)^2(k+1)^2|\mathcal{E}| = d \times |\mathcal{E}| \left(\frac{1}{4}k^4 + 6k^3 + \mathcal{O}\left(k^2\right)\right).$$

The number of nonzeros in A is more complex. The rows of A are indexed by the various nodes. For each vertex node, v, there will be a nonzero for every node within the *star* of the vertex, i.e., the union of the triangles meeting at v. Let the number of such triangles be denoted by $N(v)$. Accounting duplicate representation for the vertex and edge nodes in the interior of the star of v, the number of nonzeros is easily seen to be

$$1 + N(v) \times \left(\frac{1}{2}(k+1)(k+2) - (k+1)\right) = 1 + N(v) \times \left(\frac{1}{2}k(k+1)\right).$$

Thus the total number of nonzero matrix entries in rows indexed by vertex nodes is equal to

$$V + \sum_v N(v) \times \left(\frac{1}{2}k(k+1)\right)$$

where V is the total number of vertices. We can evaluate $\sum_v N(v)$ as follows. Put $N(v)$ marbles at each interior vertex, then distribute them the triangles in the star of v. In this way, each interior triangle will obtain three marbles, so

$$\sum_v N(v) \approx 3|\mathcal{E}|$$

where $|\mathcal{E}|$ is the total number of triangles. Thus the total number of nonzero matrix entries in rows indexed by vertex nodes is approximately equal to

$$V + 3|\mathcal{E}| \left(\frac{1}{2}k(k+1)\right).$$

From Euler's formula, we have

$$|\mathcal{E}| - E + V = c$$

where E denotes the number of edges and c depends only on the topology of the domain. Using two marbles for each edge, we see that twice number of edges is approximately equal to $3|\mathcal{E}|$. Therefore

$$V \approx \frac{1}{2}|\mathcal{E}|.$$

Thus the number of nonzero matrix entries attributable to vertex nodes becomes

$$|\mathcal{E}| \left(\frac{1}{2} + \frac{3}{2}k(k+1)\right).$$

The number of edge contributions is seen to be

$$(k-1) \times ((k+1)(k+2) - (k+1))$$

times the number of edges. This is because there are $(k+1)(k+2) - (k+1)$ nodes in two adjacent triangles and $k-1$ nodes in the interior of each edge. Thus the total number of edge contributions is approximately

$$\frac{3}{2}|\mathcal{E}|(k-1)(k+1)^2.$$

Finally, the number of contributions from interior nodes is exactly

$$|\mathcal{E}|\frac{1}{4}(k-1)(k-2)(k+1)(k+2).$$

Thus the total number of all nonzeros in A is

$$d \times |\mathcal{E}| \times G(k)$$

where

$$G(k) = \frac{1}{2} + \frac{3}{2}k(k+1) + \frac{3}{2}(k-1)(k+1)^2 + \frac{1}{4}(k-1)(k-2)(k+1)(k+2).$$

Therefore the work estimates for a form evaluation for the two approaches agree exactly to order k^4. However, in computing the global matrix, one must do an extra evaluation using element matrices just to compute the global matrix. Let $L(k)$ denote the coefficient of $d \times |\mathcal{E}|$ in the work estimates for the technique using element matrices only:

$$L(k) = \frac{1}{4}(k+1)^2(k+2)^2.$$

Thus the cost of r evaluations of $a(\mathbf{u}, \mathbf{v})$ using the global approach exceeds that of the local approach when

$$rG(k) + L(k) > rL(k)$$

or equivalently when

$$r < \frac{L(k)}{L(k) - G(k)} = 1 \Big/ \left(1 - \frac{G(k)}{L(k)}\right). \tag{3.1}$$

Note that $G(k)/L(k)$ tends to one as k increases.

In the piecewise linear case, we have $G(1)/L(1) = .389$, so the global approach is cheaper as soon as $r \geq 2$. However, for $k = 4$, the global approach is more expensive for $r \leq 6$. For $k = 10$, it is more expensive for $r \leq 24$. If one is using a multilevel iterative scheme to solve the linear systems related to the bilinear form, then it can be expected that only a small number of evaluations of $a(\cdot, \cdot)$ will ever be done.

Writing $t = k + 1$, we can simplify the expressions for G and L. We find

$$L(t) = \frac{1}{4}t^4 + \frac{1}{2}t^3 + \frac{1}{4}t^2$$

and

$$\begin{aligned}
G(t) =& \frac{1}{2}+\frac{3}{2}t(t-1)+\frac{3}{2}(t-2)t^2+\frac{1}{4}(t-2)(t-3)t(t+1) \\
=& \frac{1}{2}+\frac{3}{2}t(t-1)+\frac{3}{2}(t-2)t^2+\frac{1}{4}(t^3-5t^2+6t)(t+1) \\
=& \frac{1}{2}+\frac{3}{2}(t^2-t)+\frac{3}{2}(t^3-2t^2)+\frac{1}{4}(t^4-4t^3+t^2+6t) \\
=& \frac{1}{4}t^4+\frac{1}{2}t^3-\frac{5}{4}t^2+\frac{1}{2}.
\end{aligned}$$

Thus the above expression becomes

$$r < \frac{t^4+2t^3+t^2}{6t^2-2} = \frac{(t+1)^2}{6-2t^{-2}} \approx \frac{(k+2)^2}{6}$$

where we recall that $t = k + 1$. Combining the above estimates yields the following.

THEOREM 3.1. Suppose that $d = 2$. If the number of bilinear form evaluations, r, satisfies

$$r \leq \frac{(k+2)^2}{6}$$

where k is the polynomial degree of finite elements, then it is cheaper *not* to form the global stiffness matrix, but rather to compute the forms directly by summing the local forms at each occurrence.

The three dimensional case

In the three-dimensional case, the techniques are more involved but similar in spirit. The work to evaluate $a(\mathbf{v},\mathbf{w})$ using element matrices is, for polynomials of degree k, precisely $d \times D_k^2 \times T$, where T is the number of tetrahedra in the triangulation and $D_k = (k+3)(k+2)(k+1)/6$ is the dimension of the space of polynomials of degree k in three dimensions. (We drop the common factor of $d = 3$ in all of the computations here.) Let F, E and V denote the number of faces, edges and vertices, respectively, in the triangulation. As in the two dimensional case, one relation is provided by Euler's formula:

$$T - F + E - V = c$$

where c depends only on the topology of the domain. Putting two marbles on each face and then moving each into opposing tetrahedra shows that

$$F \approx 2T.$$

Combining these two expressions shows that

$$E \approx T + V.$$

We can no longer find a simple relation between V and T. On a regular mesh based on boxes having six tetrahedra per box, we would have $T \approx 6V$. On the other hand, we could equally well have a regular mesh based on boxes having five tetrahedra per box, with $T \approx 5V$. In either case, the contribution from the vertices in the expression above for E is small.

For each vertex node, v, there will be a nonzero for every node within the *star* of the vertex, i.e., the union of the tetrahedra meeting at v. Let the number of such tetrahedra be denoted by $N_T(v)$. As in the two-dimensional case, we have

$$\sum_v N_T(v) \approx 4T$$

where T is the number of tetrahedra. Let $N_F(v)$ and $N_E(v)$ be the numbers of faces and edges meeting at v. Note that the number, $N_V(v)$, of vertices in the star of v (not including v) is equal to $N_E(v)$.

For v in the interior of Ω, the boundary of the star of v is topologically a sphere. Thus we can count the numbers of tetrahedra, faces and edges meeting at v as follows. Each tetrahedron in the star of v corresponds to a face on the boundary of the star, each face meeting at v corresponds to an edge in the corresponding triangulation of the boundary of the star, and each edge corresponds to a boundary vertex. Thus Euler's formula tells us that

$$N_T(v) - N_F(v) + N_E(v) = 2.$$

Using marbles again, on the boundary of the star only, we find

$$2N_F(v) = 3N_T(v).$$

Therefore

$$N_F(v) = \frac{3}{2}N_T(v), \qquad N_V(v) = N_E(v) = \frac{1}{2}N_T(v) + 2$$

Summing the expressions above, we find

$$\sum_v N_F(v) \approx 6T, \qquad \sum_v N_V(v) = \sum_v N_E(v) \approx 2T + 2V.$$

Suppose each edge, e, is shared by $N_T(e)$ tetrahedra. Again by using marbles on the boundary of the star of v, we find

$$\sum_{e\in\sigma(v)} N_T(e) = 3N_T(v)$$

where $\sigma(v)$ is the star of v. Moreover,

$$\sum_e N_T(e) \approx 6T$$

since

$$12T \approx 3\sum_v N_T(v) = \sum_v \sum_{e\in\sigma(v)} N_T(e) = 2\sum_e N_T(e).$$

The latter equality just says that every edge has two vertices.

The nonzero matrix entries associated with a vertex v are all those contained in the star of the vertex. To count these, we take the union of all the local nodes (excluding v) over all the tetrahedra in the star of v, then account for duplications. Nodes on each edge e will be counted $N_T(e)$ times, so we must reduce the count by $(N_T(e)-1)k$ for each e. Summing this over all edges in the star gives a reduction of $(3N_T(v) - N_E(v))\,k$. The number of redundant nodes on any face and not on the edges already considered is simply $\frac{1}{2}(k-1)(k-2)+(k-1)$ which counts the nodes in the interior of the face plus the nodes on the interior of the edge lying on the boundary of the star. Accounting for duplicate representation for the vertex, edge and face nodes in the star of v, the number of nonzeros is seen to be

$$1 + N_T(v)\times(D_k - 1) - (3N_T(v) - N_E(v))\times k - N_F(v)\times\frac{1}{2}(k-1)^2.$$

Thus the total number of nonzero matrix entries in rows indexed by vertex nodes is equal to

$$\begin{aligned} V&+4T\times(D_k-1) - 12T\times k + 2(T+V)k - 6T\times\frac{1}{2}(k-1)^2 \\ &= V\times(2k+1) + T\times\left(4D_k - 4 - 3(k-1)^2 - 12k + 2k\right) \\ &= V\times(2k+1) + T\times\left(4D_k - 3k^2 - 4k - 7\right). \end{aligned}$$

The number of edge contributions for a given edge, e, is the number of edge nodes in the interior of the edge times the number of all nodes on the edge plus, for each tetrahedron meeting there, all local nodes for a tetrahedron excluding one full face:

$$(k-1)\times\left(k+1+N_T(e)\left(D_k - \frac{1}{2}(k+1)(k+2)\right)\right).$$

Summing this over all edges, we find the total number of edge contributions is approximately

$$E(k^2-1)+(k-1)6T\left(D_k-\frac{1}{2}(k+1)(k+2)\right).$$

The number of contributions from face nodes is

$$T(k-1)(k-2)\times\left(2D_k-\frac{1}{2}(k+1)(k+2)\right)$$

(the total number of faces is about $2T$). The latter term accounts for the nodes in the two tetrahedra meeting at each face, excluding the duplications on the common face. Finally, the number of contributions from interior nodes is exactly

$$TD_{k-4}D_k.$$

Thus the total number of all nonzeros in A is

$$\begin{aligned}
&(k^2+2k)V+T\Bigg(4\times D_k-\left(3k^2+4k+7\right)\\
&\quad+(k^2-1)+(k-1)6\left(D_k-\frac{1}{2}(k+1)(k+2)\right)\\
&\quad+(k-1)(k-2)\times\left(2D_k-\frac{1}{2}(k+1)(k+2)\right)+D_{k-4}D_k\Bigg)\\
&\quad=(k^2+2k)V+T\Bigg(D_k\left(D_{k-4}+2k^2+2\right)\\
&\quad-\left(\frac{1}{2}\left(k^2-1\right)\left(k^2-2\right)+3\left(k^2-1\right)(k+2)-\left(k^2-1\right)+3k^2+4k+7\right)\Bigg)\\
&\quad=(k^2+2k)V+T\left(\left(D_k\right)^2-\frac{1}{2}\left(k^4+6k^3+13k^2+2k+6\right)\right).
\end{aligned}$$

On a regular mesh based on boxes having six tetrahedra per box, we would have $T\approx 6V$. On the other hand, we could equally well have a regular mesh based on boxes having five tetrahedra per box, with $T\approx 5V$. In either case, the contribution from the k^2V term is negligible.

The comparison of the direct approach versus the global stiffness matrix is dramatic in three dimensions. For $k=1$, it is $3V+2T$ for the latter versus $16T$ for the former, or nearly a factor of eight in favor of using a global stiffness matrix. However, for $k=2$, the ratio drops to less than three, and for $k=3$ one has to do three form evaluations before using a global stiffness matrix becomes more efficient. As in the two dimensional case, the break-even point grows quadratically in k. We have ignored the work associated with the counting the vertices, so this would only degrade the estimate of efficiency of using a global stiffness matrix. Using the techniques leading to (3.1), the following is proved.

THEOREM 3.2. Suppose that $d = 3$. If the number of bilinear form evaluations, r, satisfies

$$r \leq \frac{((k+3)(k+2)(k+1))^2}{18\,(k^4+6k^3+13k^2+2k+6)}$$

where k is the polynomial degree of finite elements, then it is cheaper *not* to form the global stiffness matrix, but rather to compute the forms directly by summing the local forms at each occurrence.

4. Computational Experiments

Comparisons with exact solutions will be made using the ℓ_p norms, $\|\cdot\|_p$, on the finite-dimensional space of nodal values of finite element functions defined on Ω. More precisely, let $f = \sum_j f(z_j)\phi_j$ where $\{z_j\}$ are the nodal points and $\{\phi_j\}$ denotes the standard Lagrange basis ($\phi_i(z_j) = \delta_{ij}$, the Kronecker delta). Then

$$\|f\|_p = \left(\sum_j |f(z_j)|^p\right)^{1/p}, \; 1 \leq p < \infty, \qquad \|f\|_\infty = \max_j |f(z_j)|. \tag{4.1}$$

When f denotes a vector quantity, interpret $|f|$ as the Euclidean length of the vector.

To evaluate the effectiveness of higher order elements in `fec`, we studied fluid flow in a converging channel (a.k.a. Jeffrey-Hamel flow). Such a flow is radial and after a change of variables satisfies the ordinary differential equation

$$u'' + 4u + u^2 = 6C, \quad u(0) = u(\alpha) = 0, \tag{4.2}$$

where differentiation is with respect to the polar angle ϕ and α is the angle of convergence of the channel.

Figure 1 shows the solutions of (4.2) for $\alpha = 45^o$ and $C = 10^2, 10^3, 10^4, 10^5$. These constants correspond, roughly, to Reynolds numbers Re $= 20, 73, 238, 764$ respectively.

Given a solution u of equation (4.2), we find that

$$\mathbf{u}_e(x,y) := \nu\, \frac{u(\mathrm{atan}(y/x))}{x^2+y^2}\, \mathbf{x}, \quad \mathbf{x} = (x,y) \in \Omega \tag{4.3}$$

solves the steady Navier-Stokes equations with kinematic viscosity ν over a wedge domain Ω (cf. Landau and Lifshitz, 1959). In our experiments, Ω is the quadrilateral with vertices $\{(1,0),(2,0),(1,1),(2,2)\}$. Figure 2 shows the velocity field for Jeffrey-Hamel flow in Ω with $\nu = 1$ and $C = 10^4$ (or Re $= 238$).

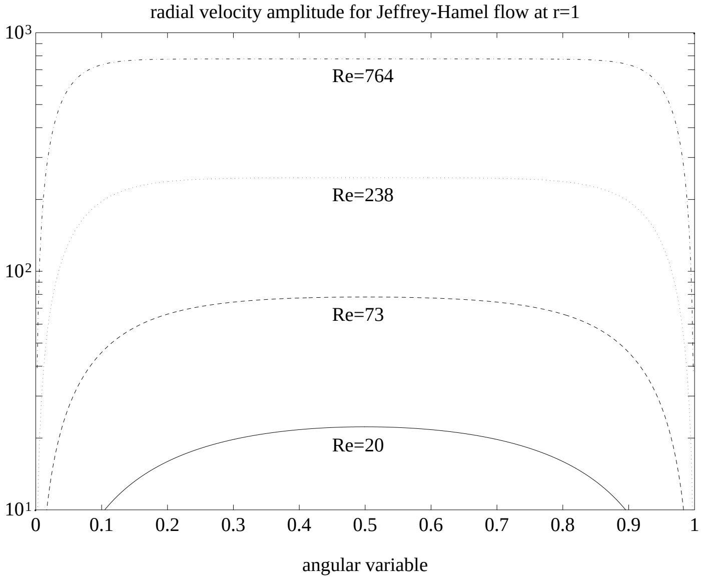


Figure 1: Solutions of equation (4.2)

In our experiments with `fec`, we solve the time-dependent Navier-Stokes equations with Dirichlet boundary conditions given by the restriction of $\mathbf{u}_e$ to the boundary of Ω. We use operator splitting and fixed point iterations to reduce the time-dependent Navier-Stokes equations to a sequence of linear Stokes problems. Representing the actions of the mass, stiffness, and convection (non-linear) operators of the Navier-Stokes equations by M, K, and C respectively, we can write the time-stepping scheme as follows. Given $u^{1,1}$, solve the following equation for $u^{n+1,k+1}$, $k \in \{1,\ldots,F\}$, $n \in \{1,\ldots,T\}$:

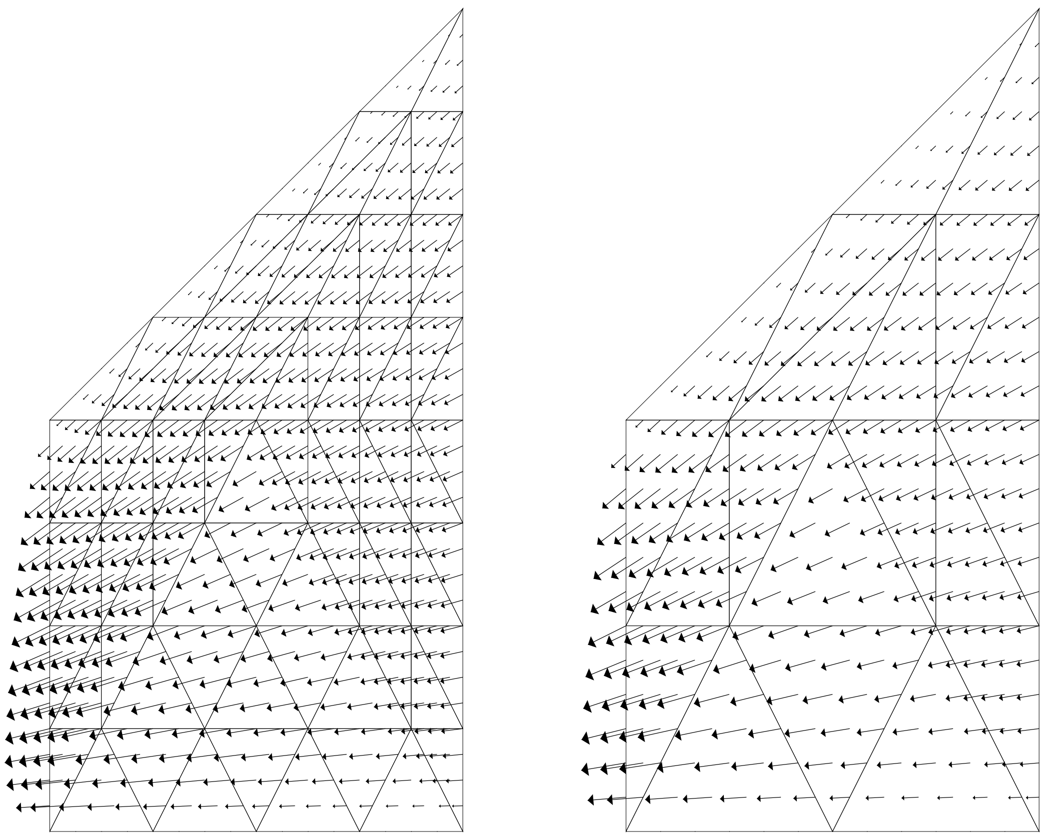
$$(\frac{1}{\Delta t}M + K)(u^{n+1,k+1}) = \frac{1}{\Delta t}M(u^n) + C(u^{n+1,k}), \quad u^{n+1,1} = u^n.$$

After F such iterations, we set $u^{n+1} = u^{n+1,F+1}$. At each of these steps, we solve F Stokes problems using the iterated penalty method (cf. Scott and Bagheri, 1990) with enough penalty iterations to ensure that the ℓ_2-norm of the divergence is no more that 10^{-10} times the ℓ_2 norm of the gradient of the velocity. For our experiments, F is fixed at 2 and T is chosen large enough to reach steady state, which was for us defined to be when $\|u^{n+1} - u^n\|_2/\Delta t < 10^{-10}$. The main step of the iterated penalty method (Scott and Bagheri, 1990) involves solving a symmetric, positive definite system, which we do by a sparse matrix method based on the minimum-degree algorithm. The system is factored only once, and from then on only forward and backward substitutions are performed.

We define the relative velocity error E to be

$$E := \frac{\|\mathbf{u}_c - \mathbf{u}_e\|_\infty}{\|\mathbf{u}_e\|_\infty},$$

where $\mathbf{u}_c$ is computed using `fec` and $\|\cdot\|_\infty$ is the norm defined in (4.1). Each experiment is characterized by three numbers: the degree of approximating polynomials p, the number of mesh subdivisions s, and the Reynolds number Re. We are interested in values of E for fixed p and varying s ("the h version") and for fixed s and varying p ("the p version").



$p = 4,\ s = 2,\ \mathrm{Re} = 238$ $\qquad$ $p = 6,\ s = 1,\ \mathrm{Re} = 238$

Figure 2: Flow in a Converging Channel

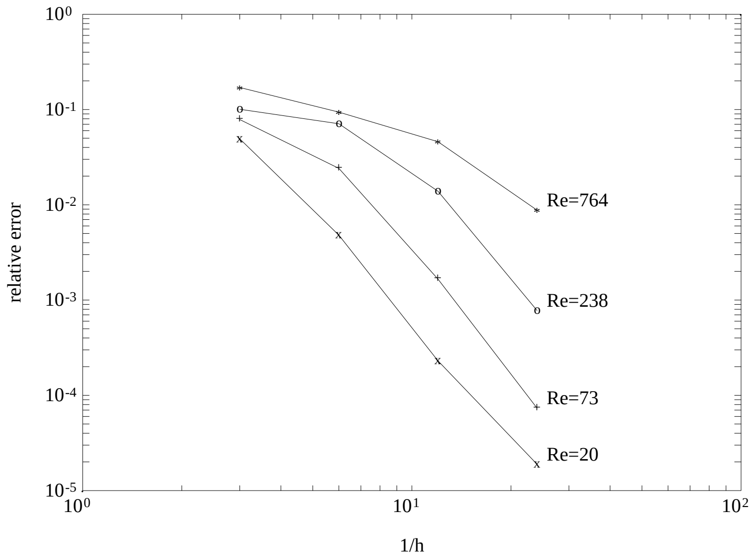


Figure 3: The h version results. Each curve plots E against s varying from 0 to 3 for $p = 4$. There is one curve for each $\mathrm{Re} \in \{20, 73, 238, 764\}$ annotated by $\{\mathrm{x}, +, *, \mathrm{o}\}$.

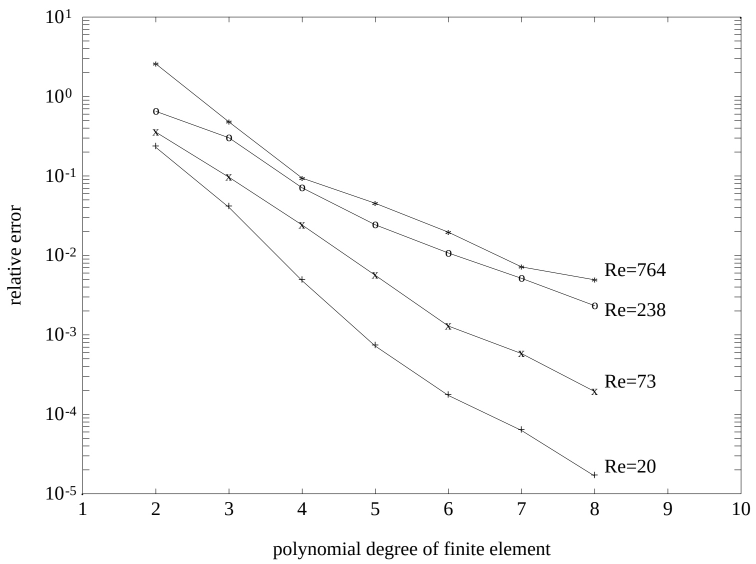


Figure 4: The p version results. Each curve plots E against p varying from 2 to 8 for $s = 1$. There is one curve for each $\mathrm{Re} \in \{20, 73, 238, 764\}$ annotated by $\{\mathrm{x}, +, *, \mathrm{o}\}$.

The main conclusion of the results in Figures 3 and 4 are that either the h method (with degree four or higher, see Scott & Vogelius 1988) or the p method are extremely accurate. It is notable that increasing the degree by only two in the p method achieves the same sort of accuracy improvement as a mesh-halving in the h method. The number of degrees of freedom is less for the p method in this case, but the associated matrix is less sparse, leading to more work in solving the linear system in the penalty iteration. In addition, the number of penalty iterations is in some cases higher for the p method (see Tables 1 and 2).

To compare the h and p versions more closely, Figure 5 plots E against the time a workstation (Sparcstation 2 with 64MB of RAM running SunOS 4.1.1 using cfront 2.1 and the native SunOS C compiler) takes to carry out one step of the time-stepping scheme. The determining factor for the work required at each time step is the number of penalty iterations needed to solve each Stokes problem. Tables 1 and 2 contain the average number of penalty iterations required in each experiment. Note that in each time step, three penalty solves are done, so that each penalty solve requires about ten iterations to achieve a relative accuracy of 10^{-10}, or about one penalty iteration per digit. The penalty parameter was chosen in each case to be equal to the parameter C in (4.2) as this heuristic seemed to keep the number of penalty iterations about constant.

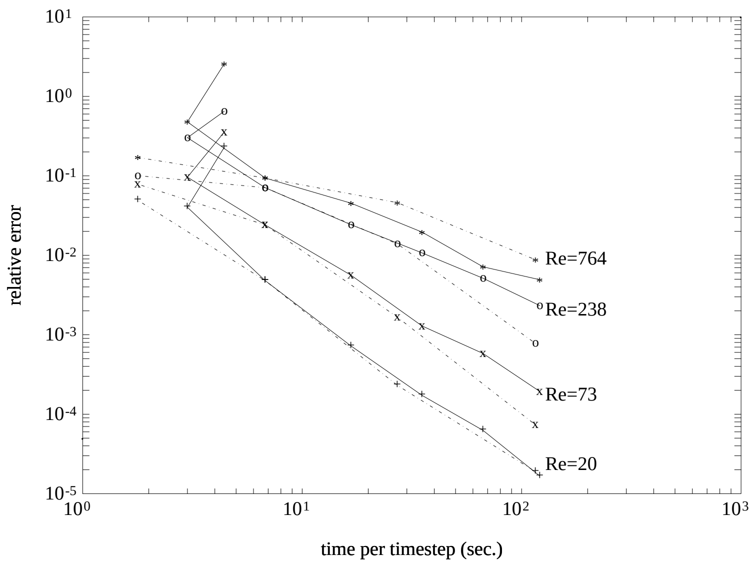


Figure 5: E vs. Time per Time-step

Note that there is a strong anomaly for degrees two and three in Figure 5, in keeping with the theory of (Scott & Vogelius 1988). This is caused by the dramatic increase in the numbers of penalty iterations (see Table 2) required due to the lack of divergence-stability for $p = 2\&3$.

Interestingly, increasing the degree of polynomials seems to be competitive with subdividing the mesh for higher Reynolds numbers. Two additional experiments, ($p = 4, s = 2, \mathrm{Re} = 1720$) and ($p = 6, s = 1, \mathrm{Re} = 1720$), produced relative velocity errors of 0.0727 (requiring 29.1 sec. per time step) and 0.0622 (requiring 33.1 sec. per time step) respectively and thus support the previous results.

	Re = 20	Re = 73	Re = 238	Re = 764
$s = 0$	49.77	187.8	29.93	17.92
$s = 1$	47	38.91	32.90	26.98
$s = 2$	44.02	47.89	29.90	26.97
$s = 3$	38.17	29.96	29.94	21.09

Table 1: Average Number of Penalty Iterations per Time Step for h Version

To investigate this point more thoroughly, we will need to extend the curves in Figure 5 to higher degree polynomials, finer meshes and higher Reynolds numbers. We will report on more extensive experiments when a parallel version of `fec` comes into production use and we are able to do larger computations.

	Re = 20	Re = 73	Re = 238	Re = 764
$p = 2$	599.9	195.7	149.7	74.98
$p = 3$	131.8	50.41	38.86	32.90
$p = 4$	47	38.91	32.90	26.98
$p = 5$	44.54	35.57	32.88	29.98
$p = 6$	44.02	35.53	32.88	26.97
$p = 7$	44	35.63	38.91	26.97
$p = 8$	140.3	35.88	38.92	26.97

Table 1: Average Number of Penalty Iterations per Time Step for p Version

Several tests were done with the code `nmg` solving the Stokes equations on a unit cube $(0,1)^3$ in three dimensions. The exact solution was taken to be

$$\mathbf{u} = 1000 * (w_y - w_z, w_z - w_x, w_x - x_y), \quad p = 10xyz$$

where $w = (x(1-x)y(1-y)(z(1-z))^2$. The appropriate right-hand side was computed in order that this yield an exact solution to the Stokes equations.

Relative ℓ_∞ errors for the velocity as defined above were computed for a range of values of s and p. These are depicted in Figure 6. In Figure 7, relative ℓ_2 errors for the pressure are depicted for a range of values of s and p.

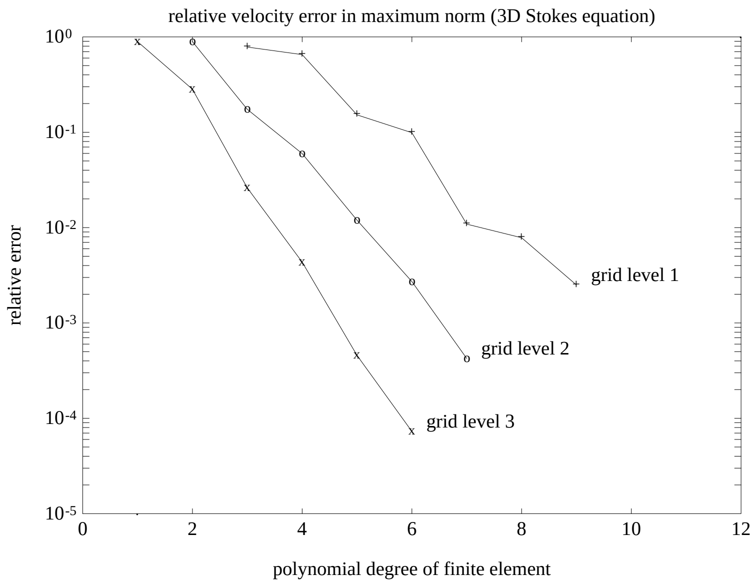


Figure 6: Velocity Errors for 3D Stokes Equations

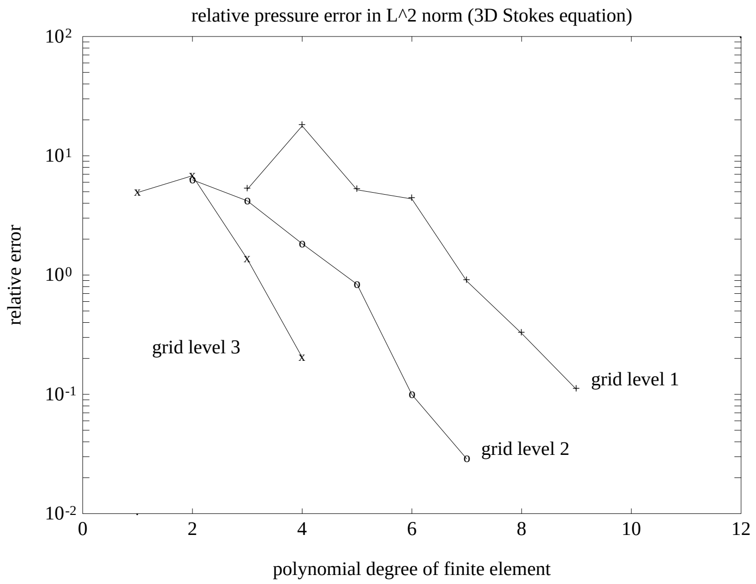


Figure 7: Pressure Errors for 3D Stokes Equations

We again see that the p method is quite effective, requiring only an increase in degree of two or three to match the effect of a mesh-having in the h method.

References

L. D. Landau and E. M. Lifshitz, **Fluid Mechanics**, Pergamon Press, 1959.

L. R. Scott & B. Bagheri, *Software Environments for the Parallel Solution of Partial Differential Equations*, **Computing Methods in Applied Sciences and Engineering IX**, R. Glowinski and A. Lichnewsky, eds., Philadelphia: SIAM, 1990, 378–392.

L. R. Scott & M. Vogelius, *Norm estimates for a maximal right inverse of the divergence operator in spaces of piecewise polynomials*, $\mathbf{M^2AN}$ **19** (1985), 111–143.

L. R. Scott & S. Zhang, *Higher Dimensional Non-nested Multigrid Methods*, **Math. Comp. 58** (1992), 457-466.